Report

General Dynamics Report
06d-ONR-01

CORE AVIONICS AND STANDARDIZATION STUDY

David L. Kellogg, J. Kirston Henderson, Mikel J. Harris, Anthony J. Schiavone

GENERAL DYNAMICS
Fort Worth Division
P.O. Box 748, Fort Worth, TX 76101

31 March 1992
Final Technical Report (CDRL Item A002) for
Contract N00014-91-C-0233

Prepared For

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5000

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION General Dynamics Corporation 6b. OFFICE SYMBOL (*If applicable*)	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (*City, State and ZIP Code*) Fort Worth Division (P.O.B.748) Fort Worth, TX 76101	7b. ADDRESS (*City, State and ZIP Code*) 800 North Quincy Street Arlington, VA 22217-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION NAV SPAWAR 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-91-C-0233
8c. ADDRESS (*City, State and ZIP Code*) Space and Naval Warfare Systems Command (SPAWAR) Washington, D.C. 20363-5109	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO.: 0604574N PROJECT NO.: TASK NO: WORK UNIT NO.
11. TITLE (*Include Security Classification*) Core Avionics and Standardization	
12. PERSONAL AUTHOR(S) Kellogg, David L.; Henderson, J. Kirston; Harris, Mike J.; Schiavone, Athony J.	
13a. TYPE OF REPORT Final	13b. TIME COVERED FROM 91/09/23 TO 92/03/31
14. DATE OF REPORT (*Yr., Mo., Day*) 92/03/31	15. PAGE COUNT
16. SUPPLEMENTARY NOTATION Prepared in cooperation with Computer Sciences Corp., IBM, and HUGHES. Approved for public release; distribution unlimited	

17 COSATI CODES			18 SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB. GR.	Avionics, Standardization
01	03	03	
14	02		

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

The standardization of avionics is based on modularity and commonality across an instruction set architecture. When budgetary cutbacks are considered, the ability to use standardization to cut costs assumes greater importance. Methodology to examine benefits of standardization is presented. Objectives associated with standards adopted by the Joint Integrated Avionics Working Group (JIAWG) and the Next Generation Computer Resources (NGCR) program are examined. Comparisons are made. Conclusions are reached. Recommendations are made.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL James G. Smith	22b TELEPHONE NUMBER (*Include Area Code*) (703) 696-4715 22c. OFFICE SYMBOL ONR Code 1211

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE

PREFACE
CORE AVIONICS AND STANDARDIZATION

As early as 1985, General Dynamics Fort Worth Division was examining Distributing Operating Systems in the context of avionics architectures. By 1988, a pre-release version of the Alpha Distributed Operating System developed under Air Force contract at Carnegie-Mellon University had been examined as an Operating System kernel for an avionics architecture. The following analysis builds on the methodology developed during that examination.

The capability of General Dynamics Fort Worth Division to examine Real-Time Operating Systems kernels was discussed with SPAWAR-231 in October, 1990. By that December, a Next Generation Computer Resources program study was being considered for the real-time reconciliation of tactical databases across multiple platforms. In preparation for such a study, General Dynamics Fort Worth Division coordinated its efforts with NOSC-413 which was drafting a white paper on the Database Management System Interface Standard for the Next Generation Computer Resources program.

By September, 1991, the General Dynamics Fort Worth Division had initiated its own six-month study of the real-time reconciliation of tactical databases across multiple platforms for the Office of Naval Research. Such an effort was to be closely coordinated with NOSC-413. However, the initiation of the NOSC-413 Database Management System Interface Standard Working Group for the Next Generation Computer Resources program was delayed. In a subsequent strategy session at SPAWAR-231, a decision was made for General Dynamics Fort Worth Division to address possible reconciliation of the POSIX Operating System and Futurebus+ Backplane standards with Core Avionics and the Joint Integrated Avionics Working Group PI-Bus Backplane standards. Accepting this challenge, the General Dynamics Fort Worth Division enlisted the help of NAVAIR-546B to interpret current status of the Navy Joint Integrated Avionics Working Group efforts and solicited additional input from the Computer Sciences Corporation Integrated Systems Division, the HUGHES Radar Systems Group's Processor Division, and the IBM Federal Sector Division. The following analysis would not be possible without the cooperation of these respective entities. The General Dynamics Fort Worth Division owes each a debt of gratitude.

ACRONYM LIST

ASAAC	Allied Standards Avionics Architecture Council
BIT	Built-in Test
CAB	Common Avionics Baseline
CIP	Common Integrated Processor
CNI	Communication Navigation Identification
CND	Can Not Duplicate
CPU	Central Processing Unit
DAIS	Digital Avionics Integrated Suite
ECS	Environmental Control System
EFA	European Fighter Aircraft
EO	Electro-Optical
EPC	European Participating Country
EW	Electronic Warfare
HOL	High Order Language
IC	Integrated Circuit
I/O	Input/Output
ISA	Instruction Set Architecture
JIAD	Joint Integrated Avionics Directorate
JIAWG	Joint Integrated Avionics Working Group
LAN	Local Area Network
LH	Light Helicopter
LRM	Line Replaceable Module
LRU	Line Replaceable Unit
MASA	Modular Avionics System Architecture Study
MMC	Modular Mission Computer
MLU	Mid Life Update
MOU	Memorandum Of Understanding
MRF	Multi-Role Fighter
MTBF	Mean Time Between Failure
MTBMa	Mean Time Between Maintenance Action
NAVAIR	Naval Air Systems Command
NGCR	Next Generation Computer Resource Program
NOSC	Naval Ocean Systems Center
OS	Operating System
PC	Personal Computer
PI-Bus	Parallel Intermodule Bus
POSIX	Portable Operating System Information Exchange
RTOK	Retest OK
RTS	Run-Time System
SEE	Software Engineering Environment
SEM-E	Standard Electronic Module E

ACRONYM LIST (Continued)

SP-Bus	Signal Processing Bus
SPAWAR	Space and Naval Warfare Systems Command
STANAGS	Standard Avionic Guide Specification
SYSCOM	System Command
TM-Bus	Test Maintenance Bus
VHSIC	Very High Speed Integrated Circuit

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

I. CHARACTERIZATION OF THE PROBLEM

I.A. INTEROPERABILITY SPACE

Any analysis of standardization requires a high degree of abstraction. Such an observation is particularly true for the objectives of standardization. Drawing on its years of experience in avionics, the General Dynamics Fort Worth Division has been able to generate observations for both SPAWAR-231 and NAVAIR-546B at a very high level of abstraction without becoming needlessly complex. Once undertaken, standardization efforts tend to lose sight of their objectives because of the massive amount of technical information they must generate. The consequences of this phenomenon are explicitly addressed in the subsequent analysis.

In the avionics application domain, the primary objective of standardization is modular interoperability (i.e., self-contained and discrete hardware "modules" used interchangeably across an architecture). Its benefits are obvious. Using interoperable modules in an aircraft (e.g., the A-X) is thought to reduce its lifecycle costs. Using interoperable modules in a variety of aircraft (e.g., the A-X and NATF) is thought to reduce lifecycle costs even further. In addition, the use of a Common Avionics Baseline (e.g., CAB III) across different types of aircraft (e.g., the F/A-18-E/F and an internationalized MRF) is also thought to reduce lifecycle costs. However, regardless of what modularity is achieved and which baseline is used, the implemented architecture must support its mission requirements as specified by the SYSCOM with oversight responsibility (i.e., NAVAIR).

In summary, modular interoperability can be approached from three different directions (reference Figure 1):

1) an Operating System comprised of interface primitives driving Instruction Set Architectures (e.g., the Mil-Std 1750A) and Backplanes (e.g., PI-Bus and Futurebus+);
2) a Tactical Database Management System comprised of access primitives driving the reconciliation and use of tactical information across an avionics architecture; and
3) Core Avionics exercising the Operating System and/or a Tactical Database Management System.

Attempting to implement modularity without understanding its effect in all three directions can be disastrous. Attempting to use commonality across avionics baselines without establishing its impact in each direction can be equally disastrous.

In the avionics application domain, modularity and commonality are not solely matters of technology, capability, performance, or logistics. Although each of these areas is clearly involved in the implementation of modularity and commonality, the overriding issue is economic because of diminishing budgets. In the face of sharp budgetary cutbacks, economic considerations assume far greater importance and demand the exploration of every opportunity for cost savings.

The overall issue of cost savings extends beyond modularity and commonality to several other closely related problem areas which must be examined (e.g., development time and cost as well as production costs). Intelligent implementations of modularity and commonality can alleviate these problem areas. Poorly conceived implementations can exacerbate them. In fact, all benefits claimed for modularity and commonality can easily be cancelled when their subsequent implementation generates more problems than it solves.

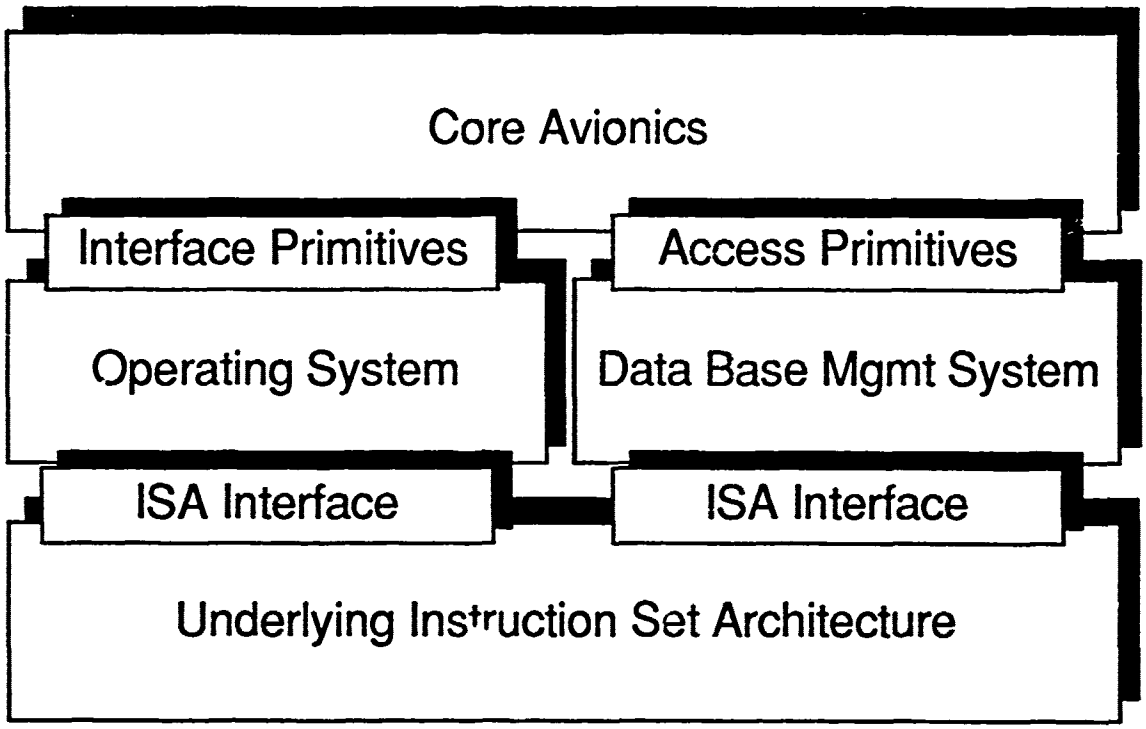


Figure 1. Modular Interoperability From an Avionics Standpoint

I.B. UNDERLYING PROBLEMS RELATED TO MODULARITY AND COMMONALITY

Because closely related problem areas should be examined before modularity and commonality efforts are undertaken, the following discussion addresses the underlying problems in current systems with respect to modularity and commonality. The discussion considers basic avionic architecture issues in the broad sense and relates them to economics. Such broad-sense issues include systems architecture, interfaces, interconnections, support, packaging, and installations. All these issues are interrelated and cannot be treated separately.

I.B.1. DEVELOPMENT TIME AND COST

Increasing the system hardware and software complexity of avionic systems continues to drive a constant increase in development cost and time. Because virtually all hardware and software for each new system is typically a new development, program schedules lengthen and costs increase as a consequence of the growing complexity. Without major technical breakthroughs, the only apparent route to significant schedule and cost reduction appears to be via significant re-use of hardware and software designs across multiple programs.

Unfortunately, little re-use of either hardware or software has been realized across different weapon systems or even within different portions of the same weapon system. Counter to several attempts to alter this situation, the general tendency has been to re-invent each hardware and software item as new application requirements are specified.

There are several fundamental obstacles to re-use. From a technical standpoint, hardware or software from one system rarely adapts easily to another system because of different operational and/or application requirements. One size simply does not fit all. The problem is further exacerbated by a continually changing technology that allows system functions to be implemented with improved hardware as each successive system is initiated. Another complication is that new

contractors introduce different viewpoints and seldom consider use of their competitor's product. Even congressional mandates appear ineffective against such apparently impregnable re-use obstacles!

If any significant degree of design commonality is to be achieved, the re-use of previous designs must be realized. An effective means to overcome re-use obstacles must be devised. Each problem must be carefully addressed and realistic solutions must be developed. Otherwise, any commonality initiative will be short-lived and ineffective.

I.B.2. PRODUCTION COSTS

Avionic systems have become a major segment of production cost for modern weapon systems and this cost element is tending to increase with each new system. Unfortunately, the basic avionics cost is only the tip of the cost iceberg. The combined costs of

(a) the closely-related production cost for electrical wiring,
(b) intermediate and depot support equipment, and
(c) spares for both avionics and avionics support equipment

increase the total actual avionic systems cost far beyond the cost of the avionics alone.

Because total production quantities for typical avionic systems are relatively low (sometimes in the range of only a few hundred), the high production-related costs for tooling, production test equipment, and production training must be amortized over small production runs. Consequently, typical military avionics production costs run many times that of non-military electronic equipment. It is not uncommon for avionics costs to exceed the price of gold on an ounce-for-ounce basis!

The major avionics-related cost of associated aircraft electrical wiring is also following the avionics complexity and cost increases. Although serial multiplex use has provided some cost relief, avionics wiring continues to be a major cost item. For example, the electrical harnesses in the multiplexed avionics of the relatively small F-16C contain more than 19,000 wires and add 742 pounds of dead weight. (This complexity increased from less than 15,000 wires for the F-16A which had similar, but less complex, avionics.) Typically, the average manufacturing and test labor per aircraft harness conductor is approximately 0.75 manhours. Assuming a typical cost to the Government in the range of $70 per manhour places the cost of each conductor at over $50. At this rate, the total electrical wiring cost for the above cited F-16C example is nearly $1 million per airplane! This is clearly a significant avionics-related cost.

I.B.3. SUPPORT COSTS

Avionics support costs over the life of a weapon system typically far exceed the high initial avionics production costs. (These costs are also rising with increasing avionics complexity.) Hence, support costs must be carefully considered as integral elements of any modularity and commonality efforts. Major contributors to support costs are discussed below.

Spares:

Avionic spares are always a major item of cost for any system. Spares must be available at each aircraft operating site to quickly replace removed avionic units at the aircraft level. If the aircraft-level replacement unit is a large and expensive avionic unit, then a corresponding avionic unit must be available as its spare. Sufficient quantities of such spares must also be provided to fill the entire repair pipeline through the intermediate avionics shop or other avionic repair center used to return failed units to serviceable condition. In cases involving remote repair sites, significant additional numbers of spares can be required to allow for transportation lags between operating units and

repair centers. For this reason, it is usually cost effective to provide local repair capabilities for a high-cost unit.

Spares cost is one of the chief driving forces toward both

(a) a high degree of avionics modularity and
(b) multi-use avionics or commonality.

Ideally, avionic systems would consists of small, low-cost (perhaps throw-a-way) electronic modules that could be quickly replaced at the aircraft level. In addition, each such module would have many applications both in the same aircraft and across multiple aircraft, thereby minimizing the number of spare types required. Such a utopian condition would result in the lowest possible spares cost plus eliminate the expensive intermediate and depot-level repair facilities. Any modularity and commonality effort should strive toward such a condition.

Support Equipment:

Avionics support equipment generally includes aircraft-level, shop-level, and depot- or factory-level test equipment. The usual absence of standardization in avionics design both within each weapon system and across different weapon systems generally leads to largely different support equipment for each avionic system. Even in efforts directed toward use of common support equipment, costly adapters and software development are usually necessary for the different systems (thereby defeating many of the standardization objectives).

Some primary factors in avionics support equipment complexity, support equipment variations between different systems, and support equipment cost are

(a) a very large number of different interface types and
(b) large numbers of interfaces between avionic elements at all levels.

Except for a few limited examples of functional and electrical interface standardization such as MIL-STD-1553, little actual standardization exists in current systems. Current standards efforts directed at high speed data bus and standard parallel interfaces such as PI-Bus and Futurebus+ offer another important but limited level of interface standards. However, the vast majority of interfaces remain entirely untouched by current standardization efforts. Even with such standards as these, the problem of variations by different manufacturers and programs remains a major hurdle. An example of such variations is evident in the several variations of MIL-STD-1553 that have evolved either as a result of the continuing process of standard definition or as a result of individual program-driven variables.

As long as the number of interface types is large, the problem of providing test equipment at all levels will be significant and standardization will be difficult. At best, such equipment continues to require costly software as well as costly and bulky test adapters for each system test application.

The very large numbers of electrical interfaces both between avionic boxes and internal cards or modules translates to excessive test equipment complexity in terms of both software and hardware. Typical avionic circuit cards or modules have several hundred individual electrical connections, each of which must be accessed and tested by the support equipment. (In many boxes, several thousand such interconnects are used.) The test problem is further complicated by the problem of isolating problems between individual cards or modules and the often complex interconnecting backplanes. Typically, the most difficult isolation problem is that of isolating a connector pin problem, especially if the problem is intermittent. In many cases, the isolation problem is so difficult that the test equipment designers adopt the totally illogical assumption that the connections are good and subsequent design test equipment with its tests are based on this assumption.

Logistics Chain:

Each avionic item subject to repair or replacement must have a suitable logistics support chain established and maintained for the life of that item. Provisions must exist for

(a) adequate spares provisioning, stocking and supply;
(b) return of repairables to repair centers; and
(c) return of repaired items to using organizations.

These logistics chains require large numbers of personnel and facilities and are a major cost element in avionics support.

Because each different type of item must be separately treated in the logistics chain, proliferation of item types significantly increases the logistics problem and cost. The current use of different designs at different points within a single weapon system and across different weapons systems results in such proliferation. Hence, logistics chain impact should rank high in modularity and commonality considerations.

I.B.4. OPERATING EFFECTIVENESS AND COSTS

Avionic systems are a key factor in both weapon systems effectiveness and operating costs. Fully operational avionics systems are essential to most modern weapon systems. Although strong emphasis upon avionics reliability has generally resulted in improved availability despite complexity increases, the need for improvement remains urgent in terms of operating effectiveness and cost.

Support Personnel:

Large numbers of support personnel are required for avionics maintenance support at the aircraft and the avionics shop to test, trouble-shoot, and repair the complex avionic systems of most military aircraft. Significant numbers of additional personnel are also required just to maintain the complex avionics support equipment used by personnel involved in avionics test and repair. All of these personnel are, in turn, supported by an entire chain of other support personnel of all types. The overall operating cost for avionics support is of major significance.

The current proliferation of different equipment types tends to increase the number of support personnel. Personnel requirements are increased because of difficulties in cross-training personnel to effectively support systems of different design. Hence, standardization that reduces the number of equipment types offers real promise of significant reductions in support personnel.

Operational Availability:

Both operating effectiveness and operating costs are closely tied to systems operational availability. Availability is, in turn, primarily driven by basic reliability and ease of repair. Both are improvement candidates as a part of modularity and commonality efforts. Likewise, both can also be adversely affected if they are not properly considered an integral part of a modularity and commonality program. The following discussion addresses some major underlying reliability and repairability problems needing attention during modularity and commonality efforts.

New Designs Introduce New Reliability Problems:

Design of new and different system elements for each application usually introduces new reliability problems that must be overcome for each new design. With the current relatively low production quantities for most system elements, it is very difficult to achieve a fully mature design. Such low quantities reduce the feedback opportunities from the field. As a consequence, it becomes extremely difficult to develop experience databases adequate enough to ferret out reliability

problems. Typically, the only problems corrected are those that arise during formal reliability testing programs. Even with the best of reliability testing, actual field conditions that may create serious problems are often inadequately simulated.

Interconnections Pose Significant Availability Problem:

Although VHSIC technology is resulting in some reductions in avionics interbox and intercard interconnections by combining more and more functions onto single cards or modules, the interconnection problem continues to loom as a major reliability and repairability factor. This often discounted, and sometimes ignored, problem appears in the basic ways discussed below.

Effect of connections on systems reliability is subject to wide disagreement. Typically, historically-based failure data collected from field failures tends to indicate acceptably high levels of reliability for interconnections. However, analysis of

(a) the usual reporting process,
(b) other field reports from field service personnel, and
(c) accounting methods for unconfirmed problems

raise serious questions regarding the validity of such statistical connection failure rate data. In fact, conclusions drawn from such statistical data are often dramatically different from conclusions made by experienced service personnel!

Most field failure reporting methods will indicate a connection failure only if a connector or connector part is replaced. If a problem is cleared after de-mating and mating a connector, any written report will typically indicate either no confirmation of the problem or a re-test OK. In fact, a very common practice for service technicians is to re-seat circuit cards or de-mate and re-mate connectors prior to test. Such practices normally stem from practical technician experience that such actions often clear problems. Such conclusions are likely to be true because the inherent mechanical wiping actions of connector pins and sockets can easily clean contacts of contamination or corrosion products that may render a connection inoperable or intermittent. In many cases, field technicians elect not to write up known connector problems because they believe such problems are so common and so inherent in avionics that another field report of failure will not change the situation.

In most analyses of field reliability, failure reports that do not result in confirmed component failure are assumed invalid and therefore discounted in assessing equipment reliability. Most such analyses conveniently ignore the possibility of intermittent failures or connection failures cleared by field card or connector re-seating. Reports of failures not confirmed as component failures are typically attributed to maintenance errors. Consequently, "observed" field reliability often comes close to the predicted values. The vast differences in "observed" reliability and actual availability for combat are dismissed as training or personnel problems!

Table I. shows reported mean time between maintenance actions (MTBMa), could not duplicate (CND), and re-test OK (RTOK) reports for a set of avionics for a current USAF fighter aircraft for one period of time. Note that nearly 23% of all maintenance actions resulted in a CND and that of items sent to the next level of repair, nearly 26% resulted in a RTOK. Similar data examined for a much later design fighter aircraft show a very similar pattern.

Table I. Typical Avionics MTBMa, CND, & RTOK Rates

Major System	MTBMa	CND %	RTOK %
Radar	5.1	30.4	25.8
Head-up Display	17.7	37.6	25.3
INS	21.2	30.7	44.1
Armament Control	30.4	0.9	22.9
Countermeasures	31.3	13.6	9.9
Air NAV Indicator/Signal Proc.	45.4	25.3	18.7
Digital Computer	68.4	13.1	57.2
ALL AVIONICS TOTAL (Includes Fuel Control, TACAN, etc)	1.7	22.9	25.7

Source: USAF 66-1, 1980

Analysis of field failure data for a current inventory aircraft computer revealed that for 73% of all removed units, no failures were confirmed. Although personnel and software errors clearly contributed to some of the removals, intermittent connections are likely culprits in many cases. The computer in question has nearly 5,000 connector pins!

The significance of the impact of connections on avionics reliability is indicated by two published study results cited below. One U.S. Navy study of avionics failure causes reported by C. N. Bain indicated that 60% of all failures were due to cables and interconnections while only 1.5% were due to IC failures [Reference 1]. A 1986 study by Harris Corporation into potential design actions to improve MTBF indicated that of all potential changes studied, the largest single improvement in an avionics computer theoretically achievable MTBF values would result from elimination of the very large number of printed circuit card backplane connections. The information in Table II. is repeated here from a table presented in 1986 by L. R. Webster & J. M. Madar of Harris Corporation [Reference 2].

Table II. Theoretically Achievable MTBF Values

Reliability Action	Resulting MTBF (Hrs.)
Baseline System MTBF	10,912
Provide Power Supply Redundancy	12,250
Eliminate VLSI Chip Failures	19,769
Eliminate Card-Chip Connections	40,000
Eliminate P/C Connections	471,216

Improvements Are Assumed To Be Made Cumulatively in Order of Listing

Source: 1986 Proceedings-IEEE Annual R&M Symposium- p.305
L.R.Webster & M. Madar, Harris Corp.

Interfaces Present Serious Repairability Problem:

Repairability in current systems is made difficult by
(a) the very large number of interfaces and
(b) the large number of different interface types
present in typical avionics systems. Because modular approaches can easily cause these problems to become even more difficult, they must be carefully considered in modularity and commonality efforts.

Large numbers of interfaces often make it extremely difficult for the technician to isolate a problem to a specific replaceable element at either the aircraft or shop level. Even the most comprehensive BIT normally does not test interfaces or isolate interface problems: The task is simply too large to tackle with a practical BIT! A similar problem typically exists even with shop test equipment used for box internal fault isolation: Here again, the sheer number of interfaces make comprehensive tests impractical!

The above problem is confounded by the large number of different interface types encountered. It is difficult to provide adequate interface test capability for complete interface testing even in complex shop-level test stations: The problem becomes totally impractical for most aircraft-level test equipment needed for interface testing.

As a direct consequence of the above test limitations, the technician is often driven to pure guesswork with respect to locations of failures to a replaceable item. In cases of a connection problem, the situation is totally impossible! The usual end result is "shotgun" maintenance in which multiple items are replaced in the hope of correcting the failure. Such actions overload the repair chain and create demand for an excessive number of spares to support the repair pipelines. Because most avionic items are far more susceptible to physical damage during handling in the repair chain than in the aircraft, a frequent side "benefit" is additional handling induced failures.

Aircraft Wiring Can Pose Serious Availability Problems:

Aircraft wiring associated with avionics poses one of the most difficult avionics system availability problems. Wiring problems can easily appear to be difficult to isolate avionic problems that often result in repeated replacements of avionic units in futile attempts to correct problems. At best, aircraft wiring problems are not only difficult to detect but, at worst, are often difficult to isolate and repair because of
(a) large numbers of bulkhead and production break connectors and
(b) frequently difficult access problems.
Such problems are often confounded by the intermittent nature of connector problems. The total elapsed time to trouble-shoot and correct wiring problems can easily be measured in days!

Because avionics-related aircraft wiring is directly related to avionics design, impact of this aspect of the system is due full attention during any modularity and commonality effort.

I.B.5. SYSTEM UPGRADE COSTS

Most military aircraft undergo one or more avionic system upgrades during their lifetime. Such upgrades typically add or change capabilities as mission changes occur. In some cases, capabilities not practical or possible during initial design because of technology limitations are included as upgrades.

In typical cases, system upgrades involve total replacement of major portions of avionic hardware and software. The existent equipment is junked. Furthermore, system support equipment is often

replaced along with the prime mission equipment, thereby increasing the upgrade cost. Such upgrades are also reflected in major upheavals in terms of logistics and training.

Such upgrades often involve complete new systems hardware and software designs. Cost of such efforts typically exceed the development and production costs of the original systems. In the case of aircraft installation provisions and wiring, such changes involve the added expense of removing existing provisions and wiring.

Because of the virtual certainty of such upgrades and the attendant high cost, any modularity and commonality effort should attempt to ease such downstream changes. New technology insertions should be accommodated without wholesale discards and re-designs.

I.B.6. ELECTRICAL POWER AND COOLING

Avionic systems have become increasingly power hungry with increasing capability and complexity. Aircraft electrical power generation and distribution systems have experienced major growth in terms of size, mass, and power taken from propulsion systems. Because most electrical power consumed by avionics must be disposed of as heat, aircraft environmental control systems (ECSs) have grown in lockstep with the electrical systems. The ECS draws increasing amounts of engine power and adds significant mass to the airplane. Because an ECS will typically dump the heat generated into the aircraft fuel supply, fuel temperatures are raised, thereby reducing engine performance. In many instances, aircraft have reached the upper limits of the heat they can dissipate.

The power and cooling issue is important to modularity and commonality efforts because choices made during such efforts can have significant effects on power and cooling. Such choices must be carefully addressed as part of modularity and commonality efforts.

Interface Power Use:

Analysis of typical avionics power uses has shown that the majority of electrical power is consumed by communication interfaces between avionics units and between avionic cards or modules. For example, it is not uncommon for current MIL-STD-1553 connected avionics units to use a third of their electrical power for serial input/output and multiplex bus drivers. (In one extreme example, two-thirds of the electrical power used by one F-16 avionics LRU is used for serial MIL-STD-1553 bus interfaces with other equipment!) Inside-the-unit communications are also high power users. Typical parallel data bus interface drivers supporting inter-card parallel bus communications account for 25% of the total power consumption.

The avionics interface power consumption situation does not appear to be improving with advanced technology systems. For example, analysis of power usage for one proposed integrated rack containing 30 SEM-E format modules using PI-Bus and High Speed Data Bus interfaces indicated that over 70% of the 750 watts consumed by the unit was used for serial and parallel interfaces. High Speed Data Bus interface modules were found to require 35 watts per module in comparison to approximately 25 watts required for the lower-speed MIL-STD-1553 interface modules. (Higher speed serial data buses normally require even more power to assure reliable communications.) Given these circumstances, it is not surprising that cooling has become a major design consideration for integrated racks.

Because of the heavy use of electrical power by interfaces, strong emphasis upon interface power reduction is a highly attractive objective of any modularity and commonality effort. The interface power use is strongly affected by which interface standards are adopted. If power consumption is not given full consideration, new standards can easily create increased levels of power and cooling problems.

Packaging Cooling Considerations:

Primary drivers in packaging of avionic cards or modules are usually

(a) module connector size requirements and
(b) the need to contain multiple modules in line replaceable units.

Typically, a minimum module width is determined by the space required for the backplane connectors. In turn, the backplane connector sizes are controlled by module interface requirements. Module height is then determined on the basis of providing sufficient card space for card functions within the established width. In cold plate cooled equipment, a secondary consideration is the requirement for an adequate heat transfer area at the module edge. Because increasing module height is often undesirable, great emphasis is frequently concentrated on

(a) improved module edge clamping for improved heat transfer or
(b) in devising improved cold plate cooling.

Some systems have resorted to liquid or complex vapor cycle cooling schemes to provide the necessary degrees of cooling. In a few desperate situations, designers have resorted to use of hollow modules with coolant circulated inside the modules.

Figure 2. provides a general view of the usual module packaging situation that results. In fact, the JIAWG SEM-E module is a recent example of a module aspect ratio set by the above considerations. Note that the typical large width-to-height ratios tend to result in long heat flow paths from electronic components at the center of the modules which tends to complicate cooling requirements.

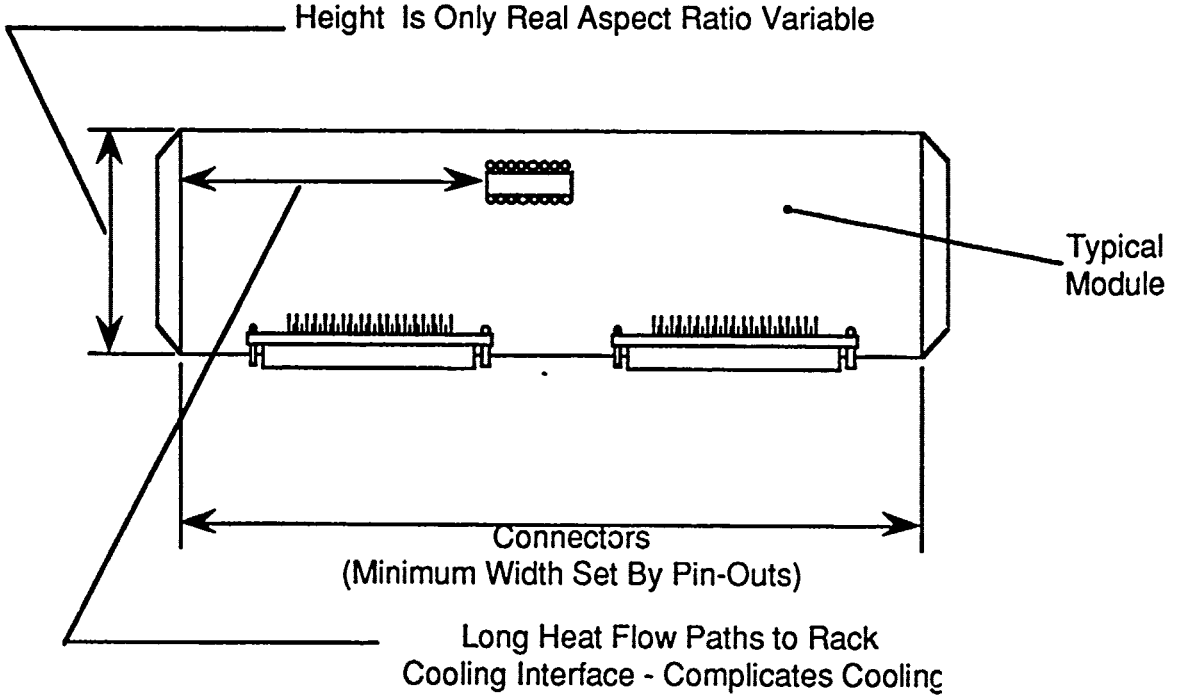


Figure 2. Typical Module Aspect Ratio Considerations

Module packaging affects the development of standards and requires careful considerations during modularity and commonality efforts. Once more, interface standards are a determining factor because interface requirements are a major driver in module packaging. Efforts toward

(a) reduction of backplane interfaces and
(b) reduction of electrical power requirements

as a part of modularity and commonality activities can have significantly beneficial effects on packaging and cooling.

I.B.7. SOFTWARE COST CONSIDERATIONS

The total lifecycle cost of system software has become a very large factor in avionic system cost. Software development is the major cost item in new systems development as well as continuing lifecycle cost item as a result of almost continual modifications throughout the life of a system. When major system hardware upgrades are incorporated, the software cycle starts over again with almost the same level of cost as the original software.

Typical avionics software programs are complex packages of many different software modules designed to run on a single or multiple set of processors. Modules may be closely related from a functional standpoint or related only by virtue of sharing a common processing function. The total complexity level and the potential impacts of timing interactions of even unrelated software processes create requirements for extensive software development testing both for initial development and for each software change. These test efforts require large amounts of manpower and elaborate software test facilities to enable validation of the software programs prior to flight use. The software test facilities hardware and simulation/test software development tasks can even exceed that of the aircraft systems development.

Software re-use between different systems of the same or different aircraft is rare for several reasons. Interfaces at the software level are again a major obstacle. Most software modules are designed with unique interfaces to other software modules, processing systems, or operating systems (each of which has unique interfaces). A particular software process or module is usually designed to execute on a particular processor and in conjunction with other software elements specifically executing on that processor or another processor. Interfaces are not uniform and each process must be designed to execute in a special system location.

The software cost situation is not hopeless and should be attacked as an integral part of any major modularity and commonality effort. Efforts toward standardization of software interfaces; reduction of non-essential interactions; development of standard modules and processes; and enabling at least some degree of transparency with regard to hardware location of software elements should be considered. Standardization of an Operating System (e.g., POSIX) or of a Backplane (e.g., Futurebus+) cannot be achieved without demonstrable impacts in Core Avionics and Tactical Database Management Systems. To achieve interoperability across an Operating System and a Backplane when it does not exist for Core Avionics and Tactical Database Management Systems will be extremely difficult.

II. CORE AVIONICS AND ARCHITECTURES

II.A. COMMON AVIONICS BASELINE

There are three major areas of consideration when choosing a common avionics baseline for both retrofit and new aircraft. The first area is the system architecture (i.e. the method of partitioning and configuring the avionics system from a box-level processing and communications perspective). The second area is concerned with the items that make up the box and how to provide the communications interfaces, data processing, and data storage to enable the box to perform its functions and communicate with the rest of the avionics system. The final area is avionics development which encompasses the tools used to develop and execute the software.

These tools include the the software engineering environment, the higher order language, the run-time support system, and the operating system.

Each area of consideration is addressed in the following sections. A brief overview of the topic is provided discussing the relevant standards and how they would apply to certain candidate aircraft. The issues are discussed for each topic, and suggestions are provided.

II.A.1. SYSTEM ARCHITECTURE

Avionic system architectures can be categorized into four generations. A first generation system architecture is one which uses entirely analog systems. This type of architecture is representative of the A-6 or the initial F-16 avionics systems.

A second generation system architecture is one in which digital avionics are incorporated into the aircraft in the form of line replaceable units. This type of architecture is representative of the avionics suites for the F-14 and F-18 aircraft and the F-16 Blocks 10-50. The second generation architecture is characterized by hardware standards such as the MIL-STD-1750A data processor; interface standards such as the MIL-STD-1553B data bus; and software standards such as the use of the Jovial J73 higher order programming language (MIL-STD-1589).

Third generation architectures apply the concept of common modules to the digital processing functions of the avionics suite. This type of architecture is representative of the avionic suites being developed for new aircraft such as the F-22 and the Comanche Light Helicopter, and retrofits being applied to existing aircraft such as the F-16 modular mission computer (MMC), where the functions of three existing avionics boxes are being replaced by a single, air-cooled box containing common modules. The F-22 uses a family of modules based on a signal processing type computer architecture, whereas the F-16 MMC uses a family of modules based on a data processing type computer architecture.

A common module based architecture encompasses a family of digital data processing, interface, and memory modules which conform to a standard size, connector and pin arrangement. The common module architecture also specifies the interfaces external to the box and the interfaces to be used for intermodule communications. A good example of a common module based architecture is provided in the Modular Avionics System Architecture Study (MASA) Final Report (FZM-8084 17 May 1991 of Contract F33657-84-C-0247 performed for the Air Force ASD-ALD/AX) where the use of common modules were investigated for retrofit of existing avionics.

Fourth generation architectures extend the common module concept to not only the digital processing functions but also the analog processing (i.e. radar, electro-optic, and electronic warfare pre-processing) functions. This architecture generation provides the specifications and building blocks for developing the entire avionics system architecture and is representative of the avionic suites being proposed for post-2000 aircraft such as the Navy's A-X and Air Force's Multi-role Fighter (MRF). Figure 3. provides a look at how existing and future Navy and Air Force aircraft are classified by their generation of avionics system architecture.

Aircraft	Generation			
	1st	2nd	3rd	4th
A-6	X			
F-14		X		
F-18		X		
AX				X
F-16 (BLK 5)	X			
F-16 (BLK 10-50)		X		
F-16 MMC		X	X	
F-22			X	
MRF				X

Figure 3. Navy and Air Force Aircraft by Avionics Architecture Generation

Issues:

Issues in the area of architecture development apply mostly to the retrofit case. For the retrofit case two options are available: replacing the functions previously performed by multiple LRU boxes into a single LRM box (such was the case for the F-16 MMC program) or replacing the entire avionics system, with supporting structure, cooling, and power modifications. Both of these options were investigated under the MASA program.

The first option, replacing the functions of multiple boxes into a single LRM box, requires investigation of the functions to be performed by the system and the cooling, volume, and other support resources available on the target aircraft. Digital data processing type functions such as the fire control, stores management, and heads-up display functions being rehosted to the F-16 MMC, can use third generation modules and a data processing type architecture while functions such as the radar, electronic warfare, and electro-optical system require a signal processing type architecture and either third generation modules only or a combination of both third and fourth generation modules. If the sensor design is to use existing pre-processing LRUs, third generation modules are required. If the sensor pre-processing LRUs are to be replaced with a modular system, then third and fourth generation modules are required.

Originally, it was thought that common module based avionics would require liquid cooling for retrofit applications. This posed a significant problem for forced air cooled aircraft such as the F-16. However, it was found that the F-16 could provide sufficient cooling for a single data processing box of this type. This may not be the case for retrofitting multiple modular avionics boxes or for third generation avionics which use an approach similar to the F-22 Common Integrated Processor (CIP).

The CIP being developed for the F-22 production program is a large, multi-tiered common module box that is based on a signal processing computer architecture and requires liquid cooling for operation. The CIP incorporates a majority of the F-22's functions. For retrofit applications under Option 1, the CIP cannot be used in its defined form for air-cooled, retrofit applications because of size and cooling constraints, but a system for performing signal processing functions for retrofit aircraft can be developed from the F-22 modules. Again, reference the MASA report for designs of signal processing systems for retrofit aircraft. If liquid cooling is still required, local liquid cooling technology is available. However, the amount of modular systems that can be retrofitted into existing aircraft is limited by the total cooling capacity of the environmental control system (ECS) unless upgrades to the cooling system are considered.

The second option, retrofitting the entire avionic suite with modular avionics, provides more flexibility to the customer but this flexibility comes at the expense of increased up-front cost. To retrofit the entire avionics system, an upgrade to the aircraft's cooling system must be considered because of the increased heat density of an entire modular avionics system, and the internal structure changed to accommodate SEM-E modular avionics racks. This need was expressed in the MASA study since a majority of the options considered overloaded the F-16 ECS and required innovative packaging techniques to fit in the unique volumes for so small an aircraft. To overcome the cooling problem, the MASA study suggested the addition of a vapor based cooling system. The other alternative that can be considered is an entirely liquid cooled based system such as that being developed for the F-22.

Suggestions:

For retrofits to existing aircraft such as the F-14, F-18, and F-16, the MASA study provides solutions from replacing single boxes in the avionics architecture to retrofitting the entire avionics system and adding new functionality. The MASA study provides trades from both a performance perspective and a cost perspective. It is suggested that functions, such as the fire control, stores management, and head-up display being rehosted for the MMC, use a CAB III data processing type architecture and that signal processing systems (i.e. radar, EO, EW) use a signal processing type architecture. It is also suggested that the F-22 CIP not be used in its defined form for the Option 1 case for retrofit of digital signal processing functions, because of the size and liquid cooling constraints. Also, the CIP should not be used for data processing only functions because of the size and overhead incurred when using a signal processing type architecture.

For retrofit of the entire avionics system using common modules (Option 2), the total cooling load for the system has to be considered. If the cooling system analysis confirms that the ECS is overloaded, hybrid systems such as the addition of a vapor based system would provide a lower cost solution than replacing the entire air-based ECS with a liquid-based ECS.

Future aircraft using fourth generation avionics will require a higher degree of functional integration than third generation designs. New aircraft such as the A-X and MRF and later versions of aircraft such as the F-22 will drive the design of the analog processing modules For development of future modular avionics systems, it is suggested that the applicable CAB specifications at the time be used. It is also suggested that the Navy be involved with the development of the analog processing modules to interject any peculiar requirements these modules may need to have for Navy systems. For a discussion of the design issues for applying modular avionics to analog pre-processing systems, see section five of the MASA final report.

II.A.2. INTERCHANGEABLE MODULES

To implement an avionics architecture based on common modules, the term common must first be defined. As of now, the term common or line replaceable modules (LRMs) embodies a host of standards developed by the Joint Integrated Avionics Working Group (JIAWG) under their Common Avionics Baseline (CAB) III specifications and modules developed for programs such as the F-22 and the Comanche Light Helicopter. These standards define things such as the number and types of modules, the interfaces between boxes and between modules, the physical attributes of the modules such as the module size and connector type, and the electrical attributes such as the connector pin assignments and power levels. Reference Figure 4.

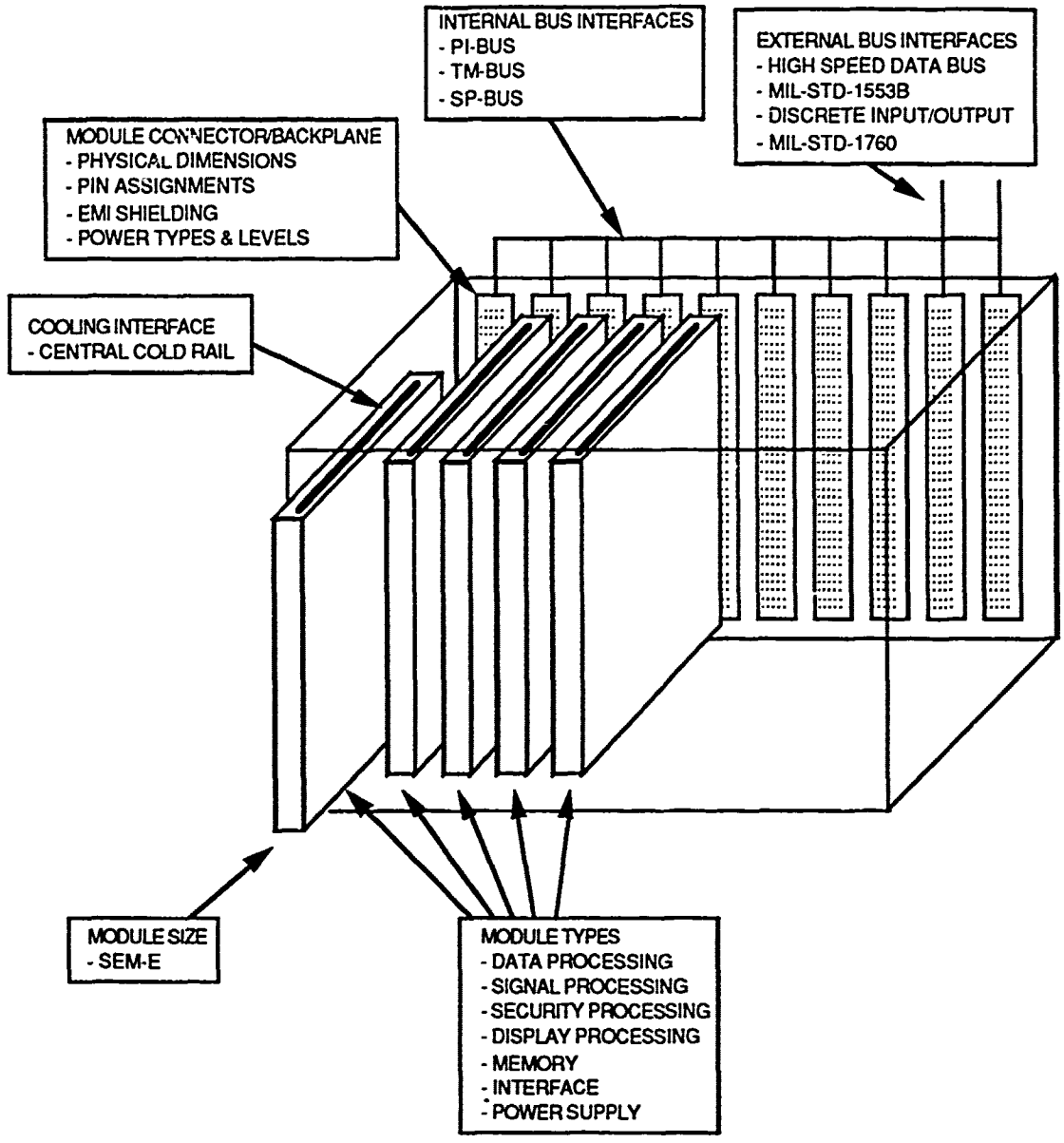
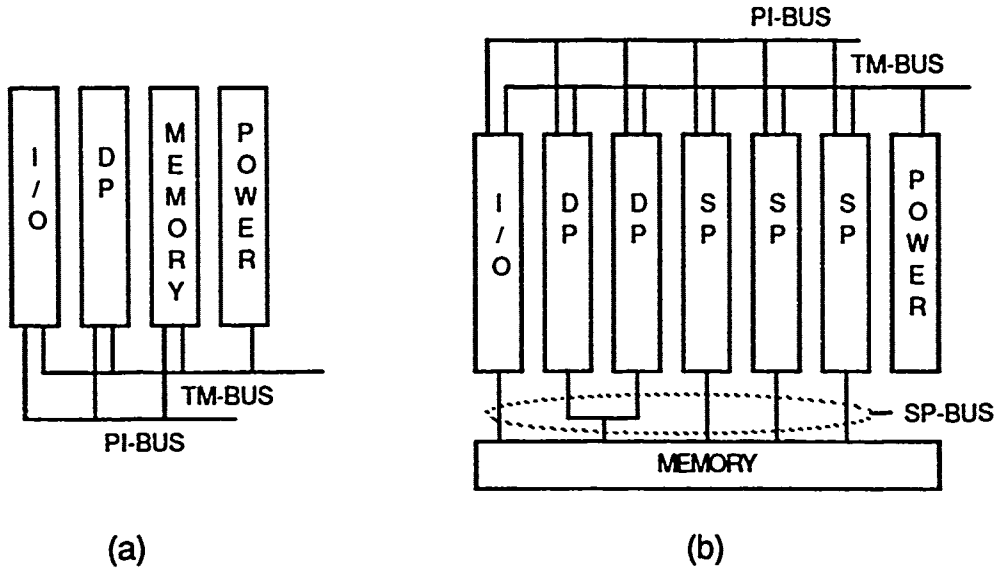


Figure 4. The JIAWG CAB III Specification Addresses all Hardware Items

The CAB III specification provides standards for almost every conceivable building block required for implementing an LRM based avionics system, with the exception of analog pre-processing modules required for the radar, E0, and EW pre-processing functions. Standards are available for implementing two different types of avionics architectures, a data processing based architecture such as the F-16 MMC, or a signal processing type architecture such as the F-22 CIP. Reference Figure 5.



Legend
I/O - Input/Output Module (e.g. 1553B, High Speed Data Bus)
DP - Data Processor (e.g. CAP16 or CAP32)
SP - Signal Processor (e.g. Fixed/Floating Point Processing Elements)
PI-BUS - Parallel Intermodule Bus
TM-BUS - Test and Maintenance Bus
SP-BUS - Signal Processing Bus

Figure 5. The CAB III Computer Architecture Types
(a) Data Processor and (b) Signal Processor

The hardware consists of both the modules required to build each system and the external and internal box interfaces. Data processing, signal processing, communications security processing, display processing, memory, interface, and power supply modules are specified and conform to the physical and electrical requirements for each architecture. External interfaces are defined for transfer of system control and status information, sensor information, video information, and weapons information. Internal interfaces are defined for transfer of information between modules (i.e. the Parallel Intermodule Bus - PI-Bus) and test and maintenance information (i.e the Test/Maintenance Bus - TM-Bus). A lot of time and money have gone into defining the standards and developing bus chip sets and module sets based on these standards. As a consequence, they should not be ignored when implementing a system based on common modules.

Areas of Concern:

The choice of an standards other than those already developed can have severe repercussions for the customer interested in developing modular avionics systems from existing common modules. A case in point is the Navy's decision to use Futurebus+ instead of the PI-Bus. Although other bus alternatives exist which outperform the PI-Bus, the choice of an intermodule bus standard must consider not only performance but also cost and the industrial base already in place with modules already based on the PI-Bus standard. The cost of developing new bus chip sets and lining up suppliers to develop modules based on a new bus standard will be high. Also, faster versions of the PI-Bus have been developed for use on the F-22 production aircraft.

Suggestions for Interchangeable Modules Standards:

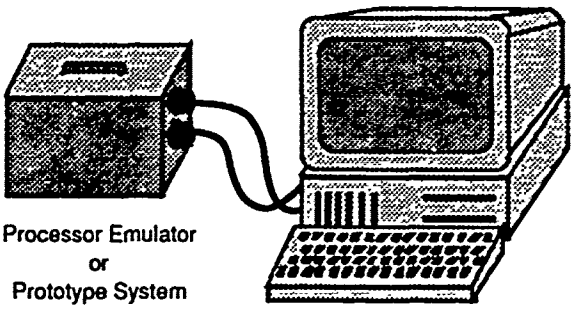
As a suggestion for choosing common module standards, the CAB III hardware specifications provide a starting point. If they are followed for the retrofit of existing aircraft, the necessary investments in time and effort will have already been made. The standard has already been defined. Bus chip sets have been developed. Module sets based on these standards are becoming available. Furthermore, a later version of CAB specifications will be used for future aircraft. It is further suggested that the Navy rethink its decision to mandate the Futurebus+ standard as an alternative to PI-Bus until Futurebus+ chip sets and common modules have been developed and are available for the Navy avionics environment.

II.A.3. APPLICATIONS PROGRAMMING INTERFACE STANDARDS

In addition to defining the hardware standards, the CAB III specification also defines the standards for software development. For software, standards can be defined for the following: the software engineering environment (SEE), the higher order language (HOL), the run-time system (RTS), the instruction set architecture (ISA), and the operating system (OS). Descriptions of these software components follow.

A SEE is a set of tools used for developing applications software. The SEE consists of both hardware and software tools. The hardware tools are things such as computer workstations and processor emulators on which the applications software will be developed and tested. The software tools execute on the computer workstations and provide the mechanism for design, development, and management of the applications software. Reference Figure 6 on the following page.

SEE HARDWARE



Processor Emulator
or
Prototype System

Computer Workstation (VAX)

SEE SOFTWARE

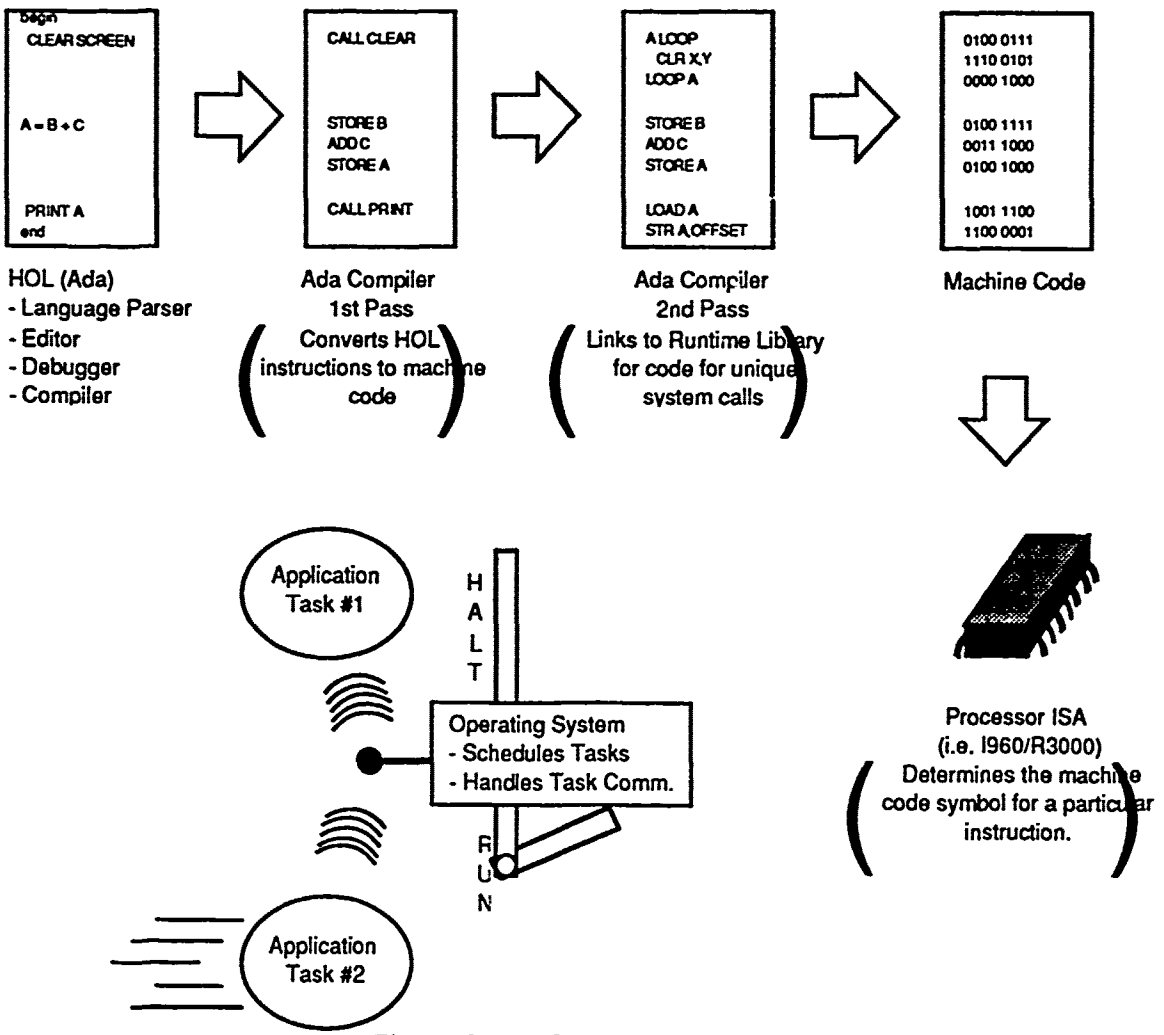


Figure 6. The Components of an SEE

A HOL is a series of commands, data structures, and methodology for developing software. The HOL includes a compiler for taking the commands (source code) and producing code to run on the target computer processor (object code). The HOL mandated for the development of new software for the Department of Defense (DoD) is Ada.

The RTS is a set of software libraries, provided by the compiler supplier, that allow the programmer to access machine specific resources. Examples of these resources are routines for reading and writing to files and storage devices; routines for reading input devices such as a keyboard or mouse; routines for graphics such as clearing the display screen and drawing lines; and system specific function calls such as accessing the time of day or date. The RTS is linked with the object code at the final stage of the compilation process.

The ISA is a lower level set of commands that the target computer processor executes. These commands are simpler than those specified in the HOL. Whereas the HOL may provide an addition operation such as A = B + C, the ISA would receive a series of commands generated by the compiler for this operation that would assign memory in the target processor for storage of the variables A, B, and C; would perform the addition operation by a series of target processor specific instructions; and would store the result in a memory location for use by the program. The instructions that the ISA executes are usually referred to as machine language.

The OS provides coordination and resources for the execution of applications software. The OS provides the resources for determining when an applications program should be executed; coordinates the communications between applications programs; and (in the avionics environment) supports the reconciliation of tactical information in hard real-time.

Areas of Concern:

Under the CAB III specification, the JIAWG specified the SEE and the ISA. The HOL was already specified as Ada by the DoD. The choice of the SEE has both advantages and disadvantages. The SEE chosen by the JIAWG uses Digital Equipment Corporation (DEC) VAX computer workstations. This is a good choice because of the wealth of support tools that interface with this platform and the Ada software development tools that can be run on this platform. The problem with this choice is that the customer is limited to equipment produced by DEC, thus providing DEC with a monopoly on the SEE for Ada software development.

The choice of an ISA also has advantages and disadvantages. The advantage of a standard ISA is that the customer is provided with an upgrade path for newer versions of processors based on the same ISA. The problems with choosing the ISA are again that the supplier has a monopoly on the chip design tying the customer to a single supplier for the processor (unless of course you pay the supplier enough money to license their designs to other companies) and the fact that this constrains the upgrade path to faster processors made (or licensed) only by that supplier. If the supplier decides to stop producing processors based on the ISA or (as a worst case scenario) goes out of business, the customer no longer has support for his products or an upgrade path to faster ISA processors.

Suggestions:

Suggestions for defining the software development environment are as follows:

(a) do not specify the type of SEE but specify contractor's responsibility to provide one in the context of a program and specify customer's SEE validation responsibility;

(b) comply with the DoD mandated Ada programming language standard;

(c) do not specify the ISA, but instead specify the interface between the HOL and the RTS (more on this below); and
(d) use the operating system provided by the supplier instead of developing your own OS since the OS is integrally tied to the hardware.

The HOL-to-RTS interface involves the definition of standard subprogram calls to be added to the HOL. As stated previously, these calls execute routines for reading and writing to files and storage devices, routines for reading input devices such as a keyboard or mouse, routines for graphics such as clearing the display screen and drawing lines, and system specific function calls such as accessing the time of day or date.

For defining this interface standard, the format and options for the commands would be standardized. The supplier of the HOL compiler would implement these subprogram calls as standard program statements and would provide the portion of the RTS that was not ISA specific. The supplier of the processor would rewrite the processor specific RTS to be compatible with the standard.

By standardizing the HOL to RTS interface, a number of advantages can be gained. First, Ada code can be machine independent; thus the same software written for an Intel 960 based data processing module can be recompiled and run on a MIPS R3000 based data processing module with either little or no software recoding required. The reason little or no software recoding is required because timing differences between the two processors may require only slight code modifications. And second, no single company has a monopoly on the processor ISA, so the customer can take advantage of competition between suppliers to produce faster, lower-cost processors.

In summary, the decision of what SEE's, ISAs, compilers, and operating systems to use for development of military avionics software should be an industrial competitive decision. It should be a program decision and not mandated by government agencies. To choose any course of action would limit the amount of competition and possibly provide a company with a monopoly on a critical component. However, before a SEE and its sets of hardware and software support tools are used for a given program, they should be validated by the contracting agency.

II.B RELIABILITY

Reliability and maintainability responsibilities include the effort required to perform the following tasks:

(a) Influencing the design of an avionics system so that it is easily maintained and, through the early use of analytical techniques, identifying as well as correcting all marginal design decisions which may have compromised its reliability;
(b) Uncovering potential maintenance problems and initiating corrective actions within a program to modify designs, specifications, and/or disassembly/assembly procedures which may compromise subsequent reliability;
(c) Performing risk reduction analyses with appropriate demonstrations which concentrate on unit reliability and module packaging within each system; and

(d) Collecting as well as documenting maintenance data from an active hardware/software testing standpoint (e.g., qualification, reliability, burn-in, simulation and acceptance) with the active support of analytical procedures.

II.B.1. TWO-LEVEL MAINTENANCE DESIGN CONCEPT

Maintainability for reliability purposes often features a two-level maintenance concept which includes the following activities:

(a) Built-in test (BIT) and autonomous self test (AST) which provide fault detection/isolation to a line replaceable module (LRM);
(b) a LRM design that allows handling, shipping, and repair without inducing failures; and
(c) sufficiently high reliability and low module costs that favors investment in spares rather than more expensive costs for intermediate maintenance facilities and personnel.

After detailed design and pricing, a level of repair analysis will always show that two-level maintenance is the most cost-effective maintenance concept.

II.B.2. AVIONICS RELIABILITY AND FUTUREBUS+

Futurebus+ is a wide parallel, high bandwidth, 2 volt collector bus. The Futurebus+ architecture resembles a memory bus where different computer components such as the processor and processor memory reside on separate modules. Avionics systems require a high level of packaging integration and functional density where volume and weight are critical considerations. For avionics, multiple modules communicate with each other using bus structures that resemble a local area network. Futurebus+ provides parity on data, address, and on some control lines. The Futurebus+ interface implementation requires over 200,000 gates, approximately 12 transceivers, and 96 to 348 active signals per module depending on the width of the data field (i.e., 32, 64, 128, or 256 bits). Currently, Futurebus+ does not provide fault tolerance, and to add fault tolerance would require additional active signals and corresponding connector pins. A dual 32-bit Futurebus+ implementation uses approximately 192 signals. A single 32-bit implementation with Error Detection and Correction would use approximately 171 signals.

Futurebus+ needs more analysis before application to avionics is possible. Some issues which need to be considered include fault tolerance, functional density, and validation and interoperability tests. Although Futurebus+ carries parity on data, address, and some control lines, other important signals are not parity protected. In addition, there are faults of state sequencing and synchroni_ation that are not detected and that can result in misleading communication.

II.C. RECONFIGURABILITY

In this section our discussion will be focused on mission critical locally distributed real-time systems. A further assumption is that the system is comprised of several types of processors (signal, general purpose, etc) interconnected by one or more local area networks and that theoretically software processes designed for a given processor type can be mapped to any processor of that type in the system. While the ability to map a process to any processor of the appropriate type in real-time is a highly desirable goal from a system availability perspective, it will only be possible in mission critical real-time systems if it can be proven in advance that the new mapping of processes to processors will satisfy all of the system critical timing requirements. In a

distributed system of even moderate complexity the number of possible combinations or mappings of processes to processors becomes astronomical very quickly.

The basic objective of a mission critical system's reconfiguration policy should be to keep alive the most critical system functions for a particular mission and mission phase. This means that we need to be able to load shed the least critical functions and the software that supports them so that those processing and local area communication resources are available to support whatever are the most critical functions at that particular time. This policy will result in the highest system availability and, therefore, the highest likelihood of mission success.

Typically, in current real-time mission critical systems, we treat all of the processes in a processor as a single process for reconfiguration purposes, i.e., we reconfigure on processor load boundaries. This results in binding software supported functions together which are both mission critical and non-critical. As was previously mentioned, criticality is dependent on mission and mission phase; therefore, functional criticality varies over time and it is not possible except in simple systems to cluster software processes together in a fixed manner based on descending criticality. Even if it were possible, it is probably not a good system availability/survivability strategy to put all of the most critical functional process eggs into one (or a few) processor basket(s). It makes more sense to reduce system vulnerability to processor failure by mapping different critical functions to different processors. So why do we typically reconfigure on processor load boundaries, if that is suboptimum? The answer lies in system testing and certification. Typically, we individually test every reconfiguration option (mapping of processes to processors) into which we will allow the system to reconfigure. How else can we prove that all of the critical system timing requirements will be met? We have not had algorithms (until recently) that will allow us to make that determination on-line and in real-time, let alone, to use as system engineering tools for designing the system in the first place except for cyclic scheduler based systems.

The cyclic scheduler approach is an ad hoc heuristic approach that fortunately works, but it produces brittle system architectures that become more and more difficult to upgrade and which do not scale well to more complex system architectures where critical system timings must be guaranteed across multiple local area bus domains.

Basically, the cyclic scheduler approach divides resource (CPU, LAN) time up into fixed slots and allocates them individually to specific processes/functions. A good mental model for this type of architecture is a mechanical clock with its gears and cogs. Everything must mesh together; therefore, this architecture does not like asynchronous events, and it does not like periodic processing streams that are not harmonic. The engineering challenge is to pound the system problem into this solution. To reconfigure in this type of architecture a new mapping of processes and messages to LAN and CPU time slots must be developed either in real-time or off-line and used as a predetermined reconfiguration option. The later approach tends to be the one used as it can be pretested and certified to work.

A better system resource (CPU, LAN, etc.) time management technique must be developed and used if we are to take full advantage of distributed system reconfiguration/availability potential. We must also develop trust in real-time reconfiguration algorithms and certify the algorithms rather than each reconfiguration option the system is allowed to operate in if we are to realize the full system availability benefits inherent in future distributed real-time mission critical systems.

II.C.1. FEDERATED VERSUS DISTRIBUTED CONTROL

The ability to reconfigure locally distributed processing and networking assets of a system depends on several high-level system resource management functions that need to be present to some degree in any distributed real-time reconfigurable systems.

These system level resource management services include

(a) Reconfiguration Management which orchestrates system reconfigurations,
(b) Initial Program Load which oversees the start-up of individual processors,
(c) Status Monitoring for individual system processing and networking assets, and
(d) Status Keeping and reporting.

A fifth system resource management service, Fault Localization, is associated with isolating and repairing system assets and returning them to service.

With the possible exception of Fault Localization all of the other system resource management functions are generally involved in the real-time reconfiguration of a system. Status Monitoring provides the information with respect to what reconfigurable (hopefully) system asset has failed; Status Keeping lets us know where a possible spare resource may be and, if there is no spare, where a less critical function may reside that could be sacrificed to free up a reconfigurable asset for a more critical function, the Initial Program Load function can then be invoked to get the right set of functionality mapped onto the system resource being reconfigured.

In order to address the issue of "Federated" versus "Distributed" control of reconfiguration (and of the system at large) we must first define what these terms imply and that is the first problem because these terms mean different things to different people. The term "Federated" really relates to how a distributed processing system should be governed or govern itself. Hence, we are entering the realm of processor system political science. The forms of human government do provide a legitimate model for discussing the forms of government for computer systems (and there had better be a consistent form of government in a complex system or there will be chaos). We will, therefore, define "Federation" in that context as it is the only one that is meaningful. A federation is a form of government wherein the individual entities surrender their sovereignty to a central authority while retaining limited residual powers of government. This maps well to many real-time distributed systems in existence today; the processors have their own operating systems for local control and the system management functions are implemented as a central authority applications on one or more of the processors. (Often these central authority system management applications are replicated for reasons of system reliability/availability.) In a larger context, we often see tiered federated systems where each subsystem is a federation. More often, we see confederations; groups of systems or subsystems which have agreed to cooperate to achieve a common purpose but which have not surrendered any sovereignty with respect to overall system management.

The term "Distributed," as it applies to processing system control or government is extremely ambiguous. Everyone knows what it means and all the definitions are different. Furthermore, the definitions change over time. "Distributed" is really not a form of government (unless anarchy, no government, is implied). What is generally meant by "distributed" is "how" a common and consistent set of central governmental resource control/management policies are to be implemented in a locally distributed system environment. "Policies" are defined as "the high level overall plans embracing the general goals and acceptable procedures, especially, of a governmental body." Will they be embodied in a centralized set of processes in a few processors or will they be "distributed"

across many or all processors, but in either case, no matter the degree of distribution, the basic system reconfiguration management policy should be invariant.

Henceforth, in this section, "federated" will be used to define an implementation approach for carrying out a set of system resource reconfiguration management policies wherein the implementer or director of those policies is a centralized entity in the system with a backup each of which is resident on separate processors. "Distributed" will be used to define an implementation approach for carrying out the same set of system reconfiguration management policies wherein the implementers of those policies is a cooperating set of processes (perhaps one in each system processor) distributed across the system's communication network.

A further assumption is that we are addressing systems which have stringent timing requirements and hard deadlines and which have to go through a stringent set of system certification/acceptance tests involving proof that the systems real-time response requirements will be met by the system design both before and after system reconfigurations. It is also assumed that future major system upgrades will be subjected to the same set of system certification/acceptance tests. If system timing is not of key concern, then the system implementers should choose whichever reconfiguration control approach (distributed or federated) yields the lowest cost and is consistent with other system design goals/requirements.

If system timing is a key concern, then there are three issues with which we must deal in establishing a new system reconfiguration:

1) Is there a viable system reconfiguration option (mapping of processes onto processors, e.g.) by which all of the systems functionality may be supported or, if not, what reconfiguration option supports those functions which are most critical to the current mission phase,
2) How much disturbance will the reconfiguration cause to ongoing processing not directly associated with the event (failure or system mode change) that is causing the need for reconfiguration, and
3) What will the systems vulnerability be to the next failure assuming no repair of currently failed system assets.

In a system with critical timing requirements, none of the above issues can be dealt with unless there is a way to predict at what levels of processor and communication resource utilization timing failures will begin to occur. A reconfiguration option is not valid if critical timing requirements will not be met. If viable reconfiguration options can be determined, they can be examined to find the one with least disturbance, or the one which will yield the least vulnerability to critical functions when the next reconfiguration is required due to a hardware failure. Note that the reconfiguration option that provides the least disturbance may also be the one that provides the greatest vulnerability to the next failure. If the system is not in a critical mission phase it may be better to accept more reconfiguration disturbance to reduce future vulnerability. Conversely, if the system is in a critical mission phase it may be better to minimize disturbance and accept more vulnerability. Operator inputs are the best way to resolve this inherent conflict.

As was previously mentioned, the number of reconfiguration options, or possible mappings of processes to processors can become astronomically large very quickly in a distributed environment. From a critical function availability perspective that is just what we want, unfortunately, there remains the practical question of how we test all of those options or prove that they will not fail to meet timing requirements. We simply cannot afford to test them all; therefore, our system time management science must allow us to prove that they will be viable

reconfiguration options. Fundamentally, there are two approaches: one old and a newer one still in the process of being developed. The old approach is the cyclic scheduler approach discussed above where the processor or bus resource is time slotted and the slots are pre-allocated. A new allocation can be determined in real-time, but that allocation will remain in effect until a new one is made. Many systems have been built using this approach, but as indicated above it does not scale well to more complex distributed systems.

The new approach based on rate or deadline monotonic scheduling seeks to use closed form mathematical equations to determine reconfiguration feasibility and it does promise to scale well to more complex distributed systems given that the resource managers in those systems adhere to its principals. Unfortunately, the problem is not completely solved for the distributed case today; however, it continues to provide the best hope for being able to analytically determine whether complex distributed systems will meet their timing requirements. There are many preemptive priority systems fielded today which conform closely to rate monotonic principals that attest to the effectiveness of the technique.

The current rate monotonic scheduling research at Carnegie Mellon University and the Software Engineering Institute has its basis in the search for real-time reconfiguration technique for reconfiguring the processing and local area networking assets for the AN/BSY-1 Submarine Combat System. The technique was used successfully in the AN/BSY-1 system to determine feasible reconfiguration options for multiple processors across three interconnected local area network domains. The rate monotonic scheduling technique is supported by the processing and network resource managers and the ability to analytically find reconfiguration options with resource utilization approaching 90% while still meeting timing requirements works in the AN/BSY-1 Combat System. What is missing today is the generalized set of equations for the fully distributed case. Dr. John Lehoczky at Carnegie Mellon University is continuing his research to extend the uniprocessor results/equations to this area. Early results are available.

With the desire to leverage future commercial technology to reduce the hardware and software costs associated with future mission critical systems where possible, it is important to influence the standards and, thereby, the commercial designs such that they provide support the type of system resource management needed, time driven resource management, to attain the responsiveness and system timing predictability required by real-time distributed mission critical systems. The Ada and Futurebus+ standards have moved to support the rate monotonic scheduling approach and POSIX appears to be moving in that direction. The Software Engineering Institute working with the NGCR program and with industry support has been very effective in making this happen.

The answer as to whether reconfiguration management should be exercised from a federated (centralized) context or from a distributed context depends to a large degree on the complexity of the system. Effectively and efficiently implementing a "distributed" system management scheme for complex distributed systems is far more difficult than a "federated" approach. Therefore, the natural progression should be to solve the management problem first using a federated approach for implementing the overall reconfiguration management policies. When that is successful, a distributed approach should be investigated.

It would seem that there is an inherent underlying assumption that the distributed reconfiguration control approach will provide some additional benefit that will justify the additional complexity and significantly greater system overhead that will surely ensue. That assumption is probably not justified from a system availability analysis perspective except for extreme cases where system availability in excess of .999 999 is required, if then. A federated resource manager can be resurrected again and again and again, down to the last remaining processor. That is a matter of system design. A distributed technique is needed to decide where to instantiate the federated reconfiguration manager and to assure that there is one and only one reconfiguration manager if it is to be capable of being resurrected on any arbitrary processor node in the system.

II.C.2. PRIORITIZATION REQUIREMENTS

If one accepts that rate monotonic scheduling techniques are also essential to support complex distributed system reconfiguration then the need for priorities as a means of implementing precedence requirements also follows. (Note that rate monotonic scheduling requires a means for eliminating unbounded resource blocking and for controlling work precedence. Preemption and priority are techniques that are used to satisfy those requirements, but they are not the only possibilities.) Given that priorities are the chosen mechanism for implementing rate monotonic scheduling in Ada and in Futurebus+ it is reasonable to assume that other technologies will utilize that technique as well.

The next question that arises then is "How many priority levels are enough?" Studies at Carnegie Mellon University have shown that for the general case a range of priorities from 64 to 256 levels is appropriate. 256 levels is approaching the ideal. The fewer the number of priority bins the more work with disparate timing requirements that gets forced into the same priority bin. Since the system is unable to distinguish between actual response time requirements within the same priority level, the utilization level at which system timing failures will occur will be less than if 256 levels were used. This means that we will not be able to get as much use out of our processing and communication hardware as would otherwise be the case. That translates to greater system cost, weight, volume, spares, etc. because we will have to have more processing and communication hardware assets to support the same required level of system performance, reconfigurability, availability, and survivability. Because we are focusing our future architectural solutions on standardized approaches, this cost will be paid many times over by the Department of Defense and the U.S. Navy if the standards do not support an adequate number of priority levels.

II.D. SCALEABILITY

The ability to increase the capacity of an architecture without requiring architectural changes is called "scaleability." When the architecture is distributed, its capacity is increased by adding to existing hardware and not replacing it. The application software remains unchanged while Operating System primitives manage the increased load. Upgrades are simple and fast. In effect, scaleability allows a system to grow beyond its original specifications. However, such growth does not occur without affecting performance response characteristics. Such characteristics, in turn, depend on the I/O bandwidth of the distributed architecture's backplane or the Local Area Network to which its hardware modules are attached. As a consequence, the I/O bandwidth of the backplane often constrains subsequent scaleability. Without belaboring the hardware implications, the following observations will concentrate on software. In particular, "scaleability" will be considered from the standpoint of software re-use.

The scheduling of real-time applications in an avionics architecture has become more difficult as the applications themselves have become more complex. Techniques of task sequencing have changed from simple cyclic scheduling (where the list of activities is executed in a rigorously defined predetermined order) to much more complex algorithms (where the order of execution changes according to a set of rules). As the orders of execution change, reusable software becomes more difficult to design and implement. Such software must be articulated at several points within the software lifecycle (e.g., the requirements phase, the design phase, and the coding phase). It is not something that just happens. It requires concerted effort by all levels in a developmental effort. From an avionics standpoint, some of the most significant requirements conflict with the writing of reusable code. Those requirements are addressed under the following topics.

Control of Interrupts:

This requirement results from real-time avionics systems having to enable and disable hardware interupts and to minimize interrupt handling latency. The facilities of a development language (e.g., Ada) do not explicitly support such latency-handling mechanisms. Therefore, avionics software becomes dependent on its underlying avionics architecture.

Control of Scheduling:

The requirement for explicit control over the scheduling and despatching regime in the avionics environment is absolutely necessary. Additionally, core avionics relies on precise and constantly predictable processing to implement the most appropriate priority scheduling of the existing avionics architecture. Since development languages (e.g., Ada) legislate against dependence on scheduling and processing priorities, avionics applications must once again depend on their underlying avionics architectures.

Critical Sections:

The requirement by embedded avionics systems to safeguard their non-interrupted execution of critical code (e.g., the digital flight control system) is essential when the code is time-sensitive. Therefore, completion of execution without interruption or preemption must be guaranteed for logically consistent execution. The development language (e.g., Ada) does not support this class of guaranteed execution and forces avionics applications to depend on their target architectures.

Cyclic Execution:

Concurrently-executing independent avionics systems depend on cyclic execution. The avionics packages are scheduled in a foreground-background environment and execute on a rigorously defined periodic basis. Suspension of tasks and preemption of lower priority programs is necessary to ensure fail-safe execution of avionics applications. In some instances, the priorities across several avionics applications might change. Because development languages like Ada avoid stipulating specifics regarding multiple program execution, cyclic execution within an avionics application may become dependent on a particular implementation and would therefore change when reused in other implementations.

Distributed Processing:

This requirement is often required in embedded avionics systems which offer a high degree of reliability and resiliency to system malfunctions. Decentralizing the system architecture increases the fault tolerance of the avionics architecture. In addition, many avionics applications are built with distribution of processing as a design criterion in order to enhance their application extensibility. However, this distribution is specific to particular system architectures which may themselves compromise code reusability.

Predictable Timing:

This requirement occurs in real-time avionics systems when a task must execute within a maximum elapsed time or distance (e.g., one second or 2,000 feet). Task initiation and completion must be accomplished within that time or distance. No development language guarantees such bounds with respect to time and distance parameters. Ironically, once realized in one avionics architecture, the same code may operate too slowly in another architecture. The configurations on one aircraft will vary greatly from the configurations on another aircraft and, thereby, make predictable timing very difficult to achieve across multiple aircraft.

Pre-Elaboration:

The requirement to minimize elaboration overhead at run-time is essential in embedded avionics systems which have specialized power-up and restart constraints. Typically, such applications attempt to demote run-time elaboration to compile-time processing by "code-staticizing" so that pre-elaboration is possible. The degree to which this can be supported is target dependent and is not at all conducive to code reusability.

Storage Control:

This requirement occurs in physically small embedded systems like avionics where storage capacity is a limited resource. In such systems, the applications software must maintain precise control over the allocation and reclamation of storage resources. In the case of overlapping information stored in two places, the reconciliation process itself becomes a predictable timing problem. As already observed, development languages do not support predictable timing problems. Neither do they support storage control problems. Therefore, storage-critical code fails to be reused in different execution environments because of dissimilar storage allocation and reclamation techniques as well as wildly varying reconciliation techniques.

In summary, any software system should be optimally reusable in the following sense. An arbitrary part from such a system could be lifted intact and used for the same purpose in a totally different system. As an example, optimal reusability would ensure that the ground-to-air fire control program for an M-1 tank could be used as the air-to-ground fire control program for an RAH-66 helicopter whose job it is to eliminate tanks. Unfortunately, optimal reusability is not attainable for a number of reasons. First, a particular part may have a dependency on other parts in the program which precludes its use in a different program without those additional parts. Second, a particular part may have a dependency on an operating system primitive which would no longer be present under a different operating system. Third, a particular part may have a dependency on a particular scheduling policy which would not transfer to another system unless the same scheduling policy were present. Before reusability is realized, much work remains to be done concerning software commonality and modularity.

II.E. EXTENSIBILITY

Extensibility is the ability to enhance capabilities of a system without invalidating existing application software. Present methodology in software deployment relies on strict process control to insure system integrity. The concept of modularity as it relates to extensibility is based on support benefits and maintenance reductions centered on the introduction of smaller line replaceable modules or LRMs. These LRMs are equipped with their own on-board diagnostic software which in some cases implies the type of application software which the LRM can execute. Any software release or LRM replacement which increases aircraft capability and performance must contain an upgrade path using existing hardware. In other words, the system must have a high degree of extensibility. A measure of extensibility is number of fielded maintenance actions due to software incompatibilities. Currently, these parameters are neither measured or traced in deployed systems.

II.E.1. EXTENSIBILITY CONCEPTS

JIAWG efforts have addressed issues of standardization including the software programming language, the operating system, the instruction set architecture, hardware interfaces and the software engineering environment. Many of the CAB III standards are interrelated to form a complete set of standards for development of an modular avionic system. This form of extensive standardization may not provide the optimum solution to build a cost effective avionic suite for the target aircraft unless all of the JIAWG standards are implemented. Through the NGCR activities,

standardization has focussed on interfaces between avionic components. With the JIAWG approach, specification of a specific SEE, operating system or ISA could impose a competition problem or monopolize the avionic development of hardware LRMs and their corresponding operating systems.

There are standardization alternatives to keep extensibility high, while providing significant improvements in the avionic support and maintenance. A supplier base can be preserved with a strong standardization effort for future LRMs. By establishing interface standards between the application code and the avionic operating system, high extensibility is possible without the need to standardize on a specific instruction set, operating system, or Ada run-time support package. Current JIAWG standards are targeted toward the F-22 or LH aircraft and are subject to technology obsolescence when new generations of integrated circuits are developed. These fourth generation systems will include different partitioning options, ISAs, and other embedded features. Larger and more accurate software models will use most of throughput gains to increase aircraft accuracy and capability. Through development of an Application Interface Standard, different ISA should be selected and supported. This would not achieve the interoperability goals but interchangeability between aircraft system software could be achieved. Proper software interface standardization and management would allow each aircraft to manufacture to their unique hardware requirements. Standardization of the interface between the aircraft support electronics and the backplane could provide an opportunity for the Navy to replace entire backplanes creating interface compatibility with future processing systems. Combined, the Application Interface Standard and a removable backplane standard would allow an upgrade path to new processing technologies without redevelopment of existing application software, reducing overall weapon system development costs.

An analogy of this abstract concept is seen in the commercial PC market. The software package "Windows" serves as the interface between machine, the human, and the application code which runs on a specific ISA family. Standardization of the application software interface provides the capability of running PC software on diverse PC platforms. The Navy standards should include a similar software interface standard, above and beyond the JIAWG standards. Standardization of power, cooling, and physical size properties of the avionic enclosures provide the final link in developing a scaleable system architecture. The final avionic design suite would be independent of the processing architectures or future technologies under development. To determine the effectiveness of an Application Interface Standard, trade studies are required to examine implementation of this standard over the life cycle of an aircraft verses no standards imposed on the aircraft at all.

New Aircraft Avionic Starts: A-X Relationships

New forms of standardization provide an upgrade path for hardware without impacting existing software until capability improvements in software and hardware can be gracefully introduced into the overall avionic system. Emphasis should be placed on standardization of the weapon system architecture for the aircraft, leaving hooks for future and for new processing architectures. This implies tight control of electronic and physical signal interfaces of the planned processing architecture. Expected IOC dates for the AX system, based on Navy reports and defense appropriation estimates, clearly indicate that the CAB V specification is the likely JIAWG standard of interest for the AX avionic suite. Navy involvement in the CAB V standardization effort would have significant return on investment, influencing interface specifications as required for use on the AX.

II.E.2. OPPORTUNITIES FOR COMMONALITY AND MODULARITY

A wide range of opportunities for commonality and modularity is open both for system upgrades and new weapon system programs. Past commonality and modularity efforts have been very

limited in both their scope and objectives. These efforts have often yielded beneficial results, but their price has been high and their gains have been considerably less than desired. However, their beneficial results should encourage future commonality and modularity efforts. The good and bad experiences from past programs can be used to guide future programs. The following discussion outlines commonality and modularity opportunities; suggests possible objectives, types and levels of standardization; and recommends necessary considerations for successful commonality and modularity efforts. The discussion uses observations made on past commonality and modularity initiatives as well as major weapon system development programs.

Commonality and Modularity Relationship:

Commonality and modularity are closely related. If any widespread degree of commonality is to be achieved, it is absolutely essential that a high degree of modularity be achieved in both systems hardware and software. Modularity is essential because "one size fits all" situations are rare with avionics systems. The Air Force DAIS architecture is one notable example of a large-scale standard that did not fit all cases. It simply did not physically fit well into the relatively small F-16 aircraft. Consequently, the F-16 program was only able to implement a design based upon the DAIS system rather than the DAIS standard. In view of such experience, a primary objective of a successful commonality and modularity effort must be selection of practical levels of modularity that will allow the widest possible application of common (standard) hardware and software modules within the same system and across different systems. Standards must be adopted for hardware and software module functions and interfaces to facilitate this objective in a practical manner that fully accommodates performance, technology, procurement, and even political requirements.

Standards Must Have Broad Appeal for Success:

Commonality and modularity standards cannot be imposed on programs unless they make real sense in terms of economics, schedule, and performance to program managers. Any advantages must be clearly beneficial to all concerned or the effort will be doomed to failure. For any standard attempted, there will usually be customer and/or contractor elements with several overriding reasons why it cannot be implemented. Without clear and significant benefits, these customer and/or contractor elements will usually prevail, thereby destroying or seriously compromising the desired commonality and modularity. Attempts by the Air Force Pave Pillar Program and the subsequent Joint Integrated Avionics Working Group (i.e., JIAWG) standards effort are primary examples of this type of situation. Sufficient "overpowering" reasons to deviate from the standards were raised by enough users to reduce much of the end product to little more than lip service. The resulting JIAWG modules were similar but far from standard or common between users. In the case of JIAWG, the standardization benefits have not been sufficient to overcome the "overpowering" reasons to deviate from its standards.

Standards Development Efficiency Required:

An efficient approach to developing standards and modularity is essential to timely standards definition. The JIAWG effort which has been on-going for the past several years is having great difficulty defining a complete and final set of standards. The large, JIAWG-type committee process is too slow and cumbersome. The objectives of its committee members are so diverse that progress toward consensus is in danger of being lapped by technology advances. A future commonality and modularity program needs to establish a standards definition process involving a small number of decision makers who are able to determine real military/industrial requirements and to efficiently sort important requirements into either like-to-have, organization-unique, or product marketing categories. One possible approach would be to select a single standards development or integration contractor that would agree to opt out of any future standards product marketing. Such an approach would hold some risk for failing to consider all inputs with equal

weight, however, it would enable definition of standards in a sufficiently short time to allow their use in real programs.

Standards Timing is Critical to Implementation:

The timing of any commonality and modularity effort with respect to target weapon system programs is critical to the success of the initiative. A standardization program that does not complete its definitions in advance of new weapon system program starts is in danger of being ineffective. Real program schedules and firm contractor commitments to other program requirements (including cost) will almost always cause program managers to avoid undefined or unproven standards or basic system elements. Program managers of major weapon system programs normally abhor significant amounts of research and development concerning standards because its associated cost and schedule risks can easily destroy their programs. In view of this situation, a separate (in advance of need definition) development and test program for proposed new standards is mandatory for the success of subsequent commmonality and modularity efforts.

Competition Protection Is Necessary:

Potential for market restrictions must be avoided under federal laws governing U.S. Government procurement. Thus, commonality and modularity activities cannot be allowed to obstruct open competition. This limitation precludes selection of any particular supplier item as a standard, unless full provisions are made to allow open and fair competition by all possible suppliers. If a particular system element such as a chip, instruction set, interface, or software item developed by a particular supplier were selected as a standard under an open competition, a minimum of unrestricted licensing rights, along with necessary technical data, to all potential suppliers would be essential to comply with competitive rules. Even with the above steps, competition objections could easily arise on the basis that the original supplier's market and production status with regard to the item would create an unfair advantage. In view of this difficult situation, this area presents a major problem that must be solved to allow effective commonality and modularity.

Technology Advances Must be Accommodated:

Any commonality and modularity effort that fails to allow infusion of new technology is destined for a short life span because of several powerful forces. Improved technology typically allows performance and/or capability gains that are demanded by users. Furthermore, many avionics technology gains result in reduced size and weight which are critical factors to aircraft designers and program managers. Finally, it can be assumed that developers and suppliers of advanced technology will insist that American industry and technological advances are being stifled unless standards are transparent to technology.

The technology advancement problem is a critical consideration in the selection of a basic approach to standards. In most cases, technology transparency objectives may well be the controlling factor to the type of standard selected. Clearly, this factor, in combination with the previously discussed competition situation, appears to rule out many hard selections of specific hardware items as common use standards. Technology transparency and competition are issues that drive toward the selection of interfaces and functionality being the primary standardization candidates.

Levels and Areas of Standardization:

The levels at which standardization is attempted and the range of commonality intended for such standards are key to the probable success of commonality and modularity efforts. If the level of standardization is too large, the "one size fits all" problem previously discussed with regard to the Air Force DAIS program obstructs progress. On the other hand, standardization too narrowly defined risks excessive interface definition and technology transparency problems which diminish

its value. Standardization of modules at a very low functional level tends to create almost insurmountable interface definitions problems because of an excessive number of interfaces. Furthermore, large numbers of interfaces can pose extreme levels of difficulty in failure isolation to a replaceable module. The current JIAWG module standards with over 300 connector pins per module are examples of the potential problem. In this case, the standards definition process has proven difficult. Serious questions remain concerning fault isolation to the individual module because of large numbers of interfaces not readily amenable to comprehensive BIT testing. The fault isolation problems become especially worrisome for integrated rack installations that necessitate isolation at the aircraft level.

Neither the same level of standardization nor the same types of standards are necessarily applicable to all areas and categories of systems or system functions. A set of standards for high-speed signal processing areas are probably not appropriate for most other less demanding system applications such as display and system control functions. Hence, an attempt to define and impose a single set of standards for all applications is likely to

(a) produce a standard that really suits no function or
(b) present such an expensive overkill for many functions that it will not be used.

For example, the current JIAWG standards appear to be directed more toward processing intensive functions. Consequently, there is little real probability of these standards being applied to the larger segment of weapon system functions which simply do not warrant such power. As a consequence, weapon system functions are thus excluded and are left with few commonality and modularity choices.

Cost Issues are a Prime Consideration:

Several cost issues must be addressed as a part of a commonality and modularity definition effort. Costs required for development, production, lifecycle maintenance, and upgrades of systems should be carefully examined when selecting the objectives, levels, and ranges of standardization. Because the commonality and modularity problem is fundamentally one of economics, minimizing total cost must be one of the primary commonality and modularity considerations. Realistic cost trade studies should be conducted to evaluate the relative total costs for a wide range of different levels and types of standards for the full range of potential commonality and modularity applications. The question should be what types are most attractive from the standpoint of cost rather than accommodating commonality and modularity in the context of usual approaches which presuppose the existence of the Soviet Union. The issues to be considered should include:

- Standards development & test cost
- Resulting system hardware and software development costs
- Support equipment hardware & software development costs
- Software development and test equipment costs
- System production costs including start-up costs
- Support equipment production costs
- Support equipment installation, facility, & maintenance costs
- Spares costs for prime mission and support equipment
- Personnel cost - All levels for prime and support maintenance
- Repair or replacement costs for failed items
- Logistics chain and transportation costs - both ways
- Likely system upgrade hardware & software cost

during life-time

One key consideration in selection of standardization levels for hardware should be that of failed common module repair or throw-away. This issue is of major importance as both

(a) the number of functions placed in a single package
and
(b) the difficulty of implementing assured repairs
increase.

Resulting reductions in support equipment at all levels and elimination of half of the logistics chain offer potentially large cost savings for the throw-away approach.

Types of Standards:

A wide range of possibilities for types of standards exists. An important task for any commonality and modularity effort is to select those types which offer the most significant payoffs. Basic standards areas available for consideration are:

- Hardware Elements (System, LRU, Module, Backplane, Component)
- Instruction Set Architectures
- Operating Systems (or Executives)
- Programming Languages
- Application Software Modules
- Interfaces

Hardware standards above the component level are proven difficult to implement for several reasons, including competition and technology changes over a very wide range of uses. Such standards typically are effective only for a single system or program. Long term applications are difficult to implement.

A class of virtual hardware standardization has been realized in some cases by standardization of functions and interfaces for a hardware item, thus allowing different items of hardware from different sources and different technologies to be used on an interchangeable basis. This type of standardization really amounts to a class of interface standardization and is likely the most realizable type of standard that can be implemented on a practical basis at a module level.

Standardization of computer instruction set architecture (ISA) such as that defined by MIL-STD-1750 is necessary as a part of a processor "interface" standard if processors or processor modules are to be produced by as standard use items or used on an interchangeable basis. Such standardization is also probably necessary if standard application software modules are to be developed and used. Unfortunately, this type of ISA standardization tends to impose some difficult to accept restrictions in many applications and results in equipment that is major overkill for other applications. Consequently, deviations from this standard are rather commonplace, thus limiting its effectiveness. In addition, use of higher order languages with compilers able to produce code for various different ISAs is tending to detract from some of the original reasons for the standard ISA. In the case of the standard higher order programming languages, the important and practical standard is the programmer interface standard.

Operating system standards, where applicable, have proved to be effective levels of standards. This is especially the case when versions of the same operating system are available for a range of different processors. In this case, the operating system to applications program interfaces are the real and effective standards. (The actual operating systems are often quite different for different machines.) Here again, the key to successful standardization has been interface standardization.

From the standpoint of program development cost, software module standardization to allow re-use of software between programs is a highly attractive, but elusive and difficult objective. Application software module standards have been difficult to realize primarily because of

(a) variation of interfaces between software modules,
(b) different ISAs, and
(c) timing interdependencies between different modules sharing the same physical processing assets.

Strong commonality and modularity program emphasis should be directed toward finding a means to standardize and re-use software modules.

The key to any successful standardization of software modules probably lies in development of interface standards for software modules in much the same manner as standard operating system interfaces now enable a degree of standardization. A second area of effort is the need to reduce possible impacts of timing interdependencies between different software modules used in different situations. The ultimate situation would be to provide a software applications module environment in which would be totally uniform and transparent to individual software modules with respect to processor interface, physical processor or processor location used, and other software processes involved in the system. Such an environment should be realizable through a combination of the right operating system approach and either standard ISAs or standard interface ISA emulators resident at each processor.

The common thread in all of the above standardization considerations is interface standardization. Interface standardization (and simplification) is clearly an important (and probably essential) key to any successful commonality and modularity effort. Interface standards fall into the following general categories:

- Software interfaces
- Mechanical packaging & Cooling interfaces
- Backplane and parallel data bus interfaces
- Serial digital interfaces
- Analog signal interfaces

The question of software interfaces has been addressed. The following discussion deals with commonality and modularity considerations needed for the remaining interfaces.

Mechanical packaging and cooling interface, backplane and parallel data bus interfaces, and serial digital interfaces have been addressed by the Air Force Pave Pillar and JIAWG efforts for those standardization attempts. However, these same areas must be revisited for any new commonality and modularity effort with the goal of seeking much needed improvements as previously discussed.

The mechanical packaging and cooling interfaces should be designed to allow easy installation and minimize cooling problems. Cooling is critical to avionics reliability and often a source of problems in aircraft. The avionics cooling design should be made tolerant of ECS variations and should have some tolerance to ECS off-normal conditions and minor mechanical installation variations without near instant destruction of the avionics. The very high price in terms of weight and engine bleed that must be paid for cooling must be given full consideration in avionics design.

It is also desirable that modules for use in integrated rack applications be made sufficiently physically rugged to withstand the handling environment when not in the aircraft rack. Typically, this environment is far more deadly to the life of the avionics than the installed environment.

The current parallel bus and backplane approaches present some serious problem that should be addressed and corrected as a part of a future commonality and modularity effort. These problems are as follows:

- Significant amounts of electrical power are

required just for communications between modules. Intensive effort is needed to correct this situation in order to
(a) reduce overall electrical power use and
(b) ease the the often critical problem of cooling the electronic modules.
Alternatives such as optical bus interfaces are available to allow such reductions.

- Very large numbers of electrical interconnect pins are required between modules and backplanes. Because
  (a) the large numbers of pins effectively dictate the module aspect ratios that adversely affect cooling interfaces,
  (b) the large numbers of electrical connections may adversely affect reliability (especially in Navy applications), and
  (c) pose nearly insurmountable fault isolation problems, strong emphasis should be directed toward major reductions in the number of such pins.

The current serial data bus interface approaches for serial communications are an interface area in need of further work as a part of a future commonality and modularity effort. The current bus approaches require large amount of electrical power, require rather large amounts of electronic parts, and pose continuing obstacles to system communications that often create undesirable requirements to implement even more buses. Attention should be directed toward development of a new, lower-power, more capable serial communications method. Such development is urgently needed to allow improved system architectures that could reduce the extent of parallel data buses from the very large and very power consuming buses now in use. Techniques for lower power and higher throughput serial communications are clearly possible and have been suggested.

II.E.3 INTERNATIONAL DEVELOPMENT AND STANDARDIZATION

During the late 1980s, the United States, United Kingdom, France and Germany issued a memorandum of understanding to cooperatively develop modular avionic system architectures and avionic standards. This program is called the Allied Standards Avionics Architecture Council (ASAAC). The U.S. MOU was signed in December 1991. U.S. government leadership is focussed through the Air Force via Wright Labs, AAAS-1. The lead US company is Boeing. The goal is to achieve global industry involvement, with each country providing government participation and leadership with their respective industries to develop modular avionics. Basically, ASAAC is adopting and rewriting the JIAWG standards as a starting point. Each standard is reviewed as appropriate to reflect U.S. restrictions and regulations and potential application across multiple platforms. Figure 7. provides an overview of the Air Force standardization initiatives.

Naval Interest in Air Force Standardization

Naval Aircraft Programs	Air Force Standardization Programs			
	JIAWG	JIAD	MASA	ASAAC
F-14, F-18			X	
F-18 E/F	X	X	X	
AX	X	X		X

Figure 7. Overview of Airforce Standardization Effort

Other countries such as Belgium, the Netherlands, Norway and Denmark are proceeding with their own modular avionic program under the Mid-Life Update (MLU) Program for the F-16 avionic system modernization effort. Called the European Participating Countries (EPCs), these countries chose not to participate in ASAAC effort even though they share common roots in adapting JIAWG standards. The EPC countries plan to insert modular avionics with the cooperation of the Air Force using the Modular Mission Computer (MMC). The MMC is based on JIAWG standards and replaces and combines the functions of older conventional processing units on the F-16 A/Bs. The MLU/MMC Programs develop actual flyable hardware for fleet aircraft, whereas ASAAC develops only standards with some demonstration to be performed. ASAAC is a standards program similar to JIAWG with the focus on next generation fighters (the MRF for the U.S.), whereas the MLU/MMC programs are targeted toward fulfilling F-16 specific requirements.

ASAAC Relationship with the Navy:

The ASAAC Program has generated a series of standards called STANAGS (Standard Avionic Guide Specifications). The CAB III JIAWG standards are the basis for the STANAGS. Each country is committing $25M towards development and demonstration of these standards with the opportunity to achieve commonality across four major countries. The ASAAC Program relies heavily on industry involvement in each country. ASAAC provides a means to develop a common NATO standards for aircraft avionic architectures. Each country's industries provide, formulate, and define their own modules based on the STANAGS. The Rafale, EFA and MRF are the target aircraft of interest.

The relationship of ASAAC to the European Community is unknown as is the relationship with NATO. These relationships are yet to be defined.

CNI Commonality Between NATO and European Participating Countries:

An opportunity exists for the Navy to influence interface standards early in the development process for new European aircraft programs. There is a unique opportunity for compatibility between traditional NATO Allies and the U.S. Services. NAVAIR has the opportunity to develop communication, navigation, and identification standards compatible with future European aircraft.

The Navy has a unique opportunity to increase European commonality with minimum investment, through direct ASAAC involvement and monitoring the EPC MLU Programs. This is of specific interest in light of the Desert Storm operations where cooperation between Allied forces was essential.

MRF and AX Relationships:

The Air Force has indicated that the Multi-Role Fi[illegible]er (MRF) weapon system (an F-16 replacement) will be the ASAAC aircraft of study for [illegible].S. Air Force standardization efforts. The Navy could offer the AX avionic system for stud[illegible]luence ASAAC standards, especially in the area of CNI systems, which shares common processing systems and sensor avionics.

Relationships with Standardization Efforts: JIAD, JIAWG, and MASA:

Development of the modular avionic systems should not be limited to new aircraft starts, such as the AX or F-22. Avionic prime contractors are seeking support and maintenance benefits for existing aircraft. Using new modular technologies will improve reliability and performance for any tactical fighter. New technologies can be applied to second generation avionic architectures, such as those found on the F-14, F-15, F-16, F-18, and A-10. The Modular Avionics System Architecture (MASA) Program and the Joint Integrated Avionics Directorate, headed by the Air Force (ASD-XRF) are seeking to apply new modular technologies and avionic system architectures to retrofit systems. The Navy was involved directly with JIAWG, the Navy has a unique opportunity to seek involvement in JIAD and MASA using the AX and F-18 systems for study.

Retrofit and New Start Aircraft Direction:

The MASA Program is targeted for applying modules to retrofit aircraft, and the JIAD Program is targeted for applying modules to new-start (MRF) aircraft. Both programs have the objective to examine the use of common modular avionics on retrofit platforms other new start aircraft other than the F-22 and LH platforms. The Navy is monitoring the MASA Program, but direct involvement in demonstration planning and funding could leverage years of Air Force and industry expenditures. The MASA Program offers the Navy an open forum to industry which is essential for retrofitting existing avionic platforms. The F-18 could be offered as a platform to the MASA Program for study demonstration and analysis. The AX program could be offered to leverage modular avionic planning on the JIAD initiative. Direct involvement in both Air Force programs would save years of architecture development expenditures provide a forum to industry, and provide avenues to initiate industry studies targeted specifically for naval aviation. Figure 7. indicates which standardization efforts by the Air Force which would interest new naval aviation programs

Modularization Across Aircraft:

Insertion of modular avionics across different aircraft requires standardization programs similar to JIAWG. Using common modules between weapon system platforms and their corresponding test equipment is desirable to save carrier space and support manpower reduction. Modularization commonality between "like" components on the same aircraft is an achievable goal if strict standards are imposed early in the design process. By aligning NGCR and MASA Programs and by aligning AX and MRF avionic development financing, significant demonstration and advanced avionic standardization can be leveraged across multiple platforms. The benefits would be lower cost avionic systems which are tailored to the target platform.

III. CHARACTERIZATION OF THE SOLUTION
AMORTIZED DEVELOPMENT AND IMPLEMENTATION COSTS

Many benefits are assumed to flow from software and hardware architecture standardization. Unfortunately, these benefits will not occur unless the impact of standardization is, in fact, in concert with the intent of standardization. From previous procurements in military avionics, examples exist where well-intentioned standardization did not automatically produce the required logistics or operational enhancements. In such instances, there was technical merit to the implementation but the result did not meet expectation. When these standardization efforts were asserted, they added delays and costs to the programs in which they were mandated. As a consequence, procurement experience clearly indicates proposed architectural standards should be constantly evaluated with respect to their initial objectives as they are implemented. Experience shows implementations based on technical merit alone can be very misleading.

Navy intent is to use standardization to "open up" the software and hardware architectures at the heart of emerging weapons programs such as the A-X, the MRF, and major retrofits to F-14 and F-18 avionics. If properly executed, such initiatives have the potential for harnessing the same kinds of forces which have led to the phenomenal changes in the commercial "open systems" arena. This explosion of commercial computer technology has produced rapidly improving capability while relentlessly reducing costs. Harnessing these forces for military avionics is certainly desirable. However, it should be noted that commercial and military environments are very different. The commercial arena tends to standardize around de facto approaches rather than the evolving technologies being developed for military applications. The commercial arena also has mechanisms in place to cope with approaches whose lifespans are decidedly shorter than the product lifespans in military environments.

NAVAIR is at a critical juncture with respect to avionics standardization. In one direction is the standardization initiative orchestrated by JIAWG. In another direction is the standardization being mandated by the NGCR program for future avionics applications. In yet another direction is the constant procurement activity associated with the A-X, NATF, F-18 E/F, F-14 Advanced AYK-14, MRF, and other such programs (each of which is in a different lifecycle phase). Each of these procurement activities is impacted by an avionics standardization initiative. The magnitude of the impact will depend upon the nature of the avionics standard and its point of adoption within the respective program lifecycles. The critical issue for NAVAIR will be its constant assessment of its return on investment as each program develops and incorporates an appropriate level of avionics standardization. While technical considerations may favor one standard over another, there is no investment merit to an architectural standard which does not have a favorable impact on lifecycle costs.

There are special considerations for the military environment. Among these are

1) the system must perform its mission,
2) military systems must be ruggedized for reliability in the field environment,
3) system lifespan typically extends to decades, and
4) the volatile nature of national priorities and threat scenarios can quickly invalidate assumptions that drive standards choices.

In stark contrast to these issues is the fact that technical standards tend to age to obsolescence very rapidly. Observation of the commercial computer "chip" market exemplifies this rapid "aging" phenomena.

If cost were the only consideration, then, the obvious standardization approach for the NGCR program would be to adopt the best currently available standards and mandate their use in all new procurement and retrofit initiatives. In today's environment this would clearly indicate selection of

commercially-oriented processors and networking components. Clearly, such a move could be considered absurd by program managers whose primary concern is whether these mandated components can satisfy specific mission requirements and survive in an avionics environment. From the standpoint of program management, cost alone is not the overriding criteria for standards selection. Performance and survivability are!

The high risk climate of military procurement makes it imperative for the Navy to develop a consistent methodology that can be used to evaluate standardization efforts. An effective methodology should accommodate both a complexity of factors and a volatility in procurement/operational assumptions. Such a methodology could be developed by blending operational performance modelling with technically-constrained econometric modelling. Operational performance modelling provides a means by which the technical robustness of standards can be analyzed. Econometric modelling provides a means by which the robustness of the economic assumptions underlying a standard can be analyzed.

Adoption of specific architectural standards must be compatible with mission operation performance objectives. Furthermore, this condition must be satisfied in each avionic application. The role of an operational performance model is to serve as the vehicle for exploring the impact of uncertainties associated with each technology factor. The technical factors which should be evaluated are those that have a potential to force architectural change and affect architectural flexibility in the future. Examples of the former are 1) industrial base technology shifts; 2) operational requirements changes; and 3) supplier base volatility.

Econometric modelling is an effective tool for evaluating lifecycle costs in military programs. This is especially true because of the volatility of assumptions and rigidity of constraints so peculiar to the military environment. However, this type of modelling must be extended to incorporate a technical constraint relating to the mission capability requirement. Suitably extended, econometric modelling provides a vehicle for exploring the impacts on lifecycle costs of the economic attributes of standards choices in the context of program uncertainties. Among these factors are the following:

1) exchanging development costs for logistics costs,
2) uncertainties and delays in procurements,
3) uncertainties and changes in deployment and support scenarios, and
4) technical obsolescence.

A technique which can be used to link the technical aspects with economics aspects is to adapt Taguchi's "signal-to-noise" concept from his experimental design method. Applying this method to econometric modelling involves establishing a composite measure of the mission "goodness" and operational readiness for each standards approach. Econometric results then provide comparative lifecycle costs in relation to technical goodness of the fielded systems.

There are cost impacts associated with the imposition of standards that must be estimated and borne by the affected procurement programs. The degree to which the procuring agency (program) absorbs those costs is a function of

1) the nature of the standard,
2) the scope of its adoption (number of applications/installations), and
3) the potential lifespan of the standard.

Although these dependencies are obvious, evaluation of standards costs rests on many potentially volatile assumptions (e.g., to which procurements it will apply; to what extent it will apply; what changes host systems will undergo during their lifecycle; etc.). Two classes of costs are associated with standards. First, the government must invest in the development, or definition, of the standard. Secondly, vendors seeking to become component suppliers must invest substantially to produce implementations of the standards. The direct cost to the government for standards

development will vary in relation to the applicability of the standard outside of military procurement. If the standard has no value outside of military applications, then the government will, in effect, provide the entire investment, with the vendor portion appearing later in component procurements.

The key to successful standards selection is to demonstrate their merit in quantifiable terms that are truly significant to the Navy. This can only be the case when the standards objective is to promote commonality, interchangeability and interoperability so as to

1) maximize the number of programs and installations over which costs can be amortized,
2) maximize the degree to which these programs can absorb evolutionary change,
3) maximize the attraction for suppliers in the form of ease of competing, length of production runs, etc.

Comparative analysis of competing architectural standards is certainly not a trivial exercise. The suggested modelling approach, however, is both feasible and capable of yielding meaningful results. Perhaps just as importantly, the modelling approach tends to force early and precise attention to those issues which focus standards on their intended objective. Without such a methodology there is a strong tendency to focus standards selection on technical features to the exclusion of equally important lifecycle cost issues. The value of an architectural standards approach ought to be predictable and demonstrable in practice.

APPENDIX
LIST OF REFERENCES

1. C. N. Bain, "Aircraft Readiness Enhancement Technology" - National Defense-, pp. 42-44, Sept. 1982.

2 L. R. Webster & J. M. Madar of Harris Corporation -1986 Proceedings- IEEE Annual R&M Symposium, p. 305.